Performance Driven Facial Animation with Blendshapes

Ravikumar, Shridhar

Award date:
2018

Awarding institution:
University of Bath

Performance Driven Facial Animation with Blendshapes

submitted by

Shridhar Ravikumar

for the degree of Doctor of Philosophy

of the

University of Bath

Department of Computer Science

September 2017

Contents

List of Figures

List of Algorithms

List of Tables

Summary

In this thesis, we address some of the open challenges in the area of Performance Driven Facial Capture and Animation, specifically with the goal of improving the fidelity of the capture results and making both the Modeling and Capture stages of the animation pipeline robust, inexpensive, automated and consumer friendly. We present an overview of the process of facial animation and specifically Performance Driven Facial Animation, including the Modeling, Capture and Retargeting stages. We then discuss the existing literature in the area in detail and weigh the pros and cons of the various approaches that have been presented over the last few decades along with the differences between them. We then present, in detail, our contributions to the Modeling stage of the pipeline in the form of automating the generation of actor specific Blendshape Models from a single scan of the actor's face or alternatively from a few images of the actor's face resulting in a pipeline that is automated and inexpensive, while being inclusive of actor specific nuances. We then present our contributions in the form of our marker-based Capture pipeline that improves upon traditional marker-based systems by incorporating additional features in the form of makeup patterns which are used to train a FACS classifier that is integrated with our Blendshape weight optimization in a hybrid fashion. We show that this leads to improved results especially in areas that are otherwise challenging to capture with markers alone. We then discuss our contributions to the markerless Capture pipeline and present our approach to track an actor's face with just a monocular RBG camera. We show that our method is able to achieve realistic results in spite of the missing information inherent in the monocular input by making use of static and dynamic prior information gleaned from existing animations from accurate 3D systems. We quantitatively evaluate our results comparing it with an approach using a monocular input without our spatial constraints and show that our results are closer to the ground-truth geometry. Finally, we present our results and conclusions and discuss future directions of research.

1 Introduction

Realistic rendering and animation of human faces has been a goal within the computer graphics community for a long time, pioneered by Parke et al. in 1972 [18] and followed by many others leading to vast advances in this field. The field is very multi-disciplinary and spans techniques from computer vision, computer graphics and machine learning in order to achieve efficiency, robustness and accuracy. The difficulty inherent in achieving convincing representations, renderings and animations of human faces, combined with the revolutionary impact it will have on various fields including education, medicine, psychology, forensics, human computer interaction, virtual reality and the entertainment industry, has rightfully earned it the title of the *holy grail* of computer graphics. Immersive experiences within virtual worlds will not be complete until we are able to accurately track and render human bodies and faces and visualize digital avatars in a convincing fashion. Doing so in such a way that it is suitable and affordable for consumer level applications is also important for the widespread adoption of these technologies.

Capturing and displaying the geometry, appearance and motion dynamics of the human face is a very challenging problem owing to many reasons. The human face is an extremely complex biomechanical system that is very difficult to model. It has many degrees of freedom and is capable of conveying emotions through very subtle motions and the exact control mechanisms of the motions is not known to us [2]. The human skin has unique reflective properties and these are very difficult to simulate accurately. It is a multilayer, anisotropic, viscoelastic tissue, whose mechanical behavior is dominated by collagen fibers present in the dermis [19]. Hence, accurate simulation of skin folding requires a complex volumetric representation with carefully chosen model parameters [6]. Compounding this problem furthermore, humans are trained since birth to detect the slightest facial nuance making them extremely sensitive to any irregularities in both the appearance and motion of faces. Owing to this exposure to the variations and subtleties in faces, a curious phenomenon known as the Uncanny Valley [20] occurs, where the disparity of a near perfect facial model elicits an unsettling and creepy aesthetic. A person's response to the face shifts from empathy to revulsion as it approaches, but fails to attain, a lifelike appearance. Examples of this effect can be seen in movies like The Polar Express [1] and Beowulf [21], even in spite of the many man-hours of post-processing by skilled and trained 3D artists.

A further constraint is that many of the most accomplished methods for facial animation are proprietary owned in spite of this being a field where industry and academia work symbiotically. The process of facial capture and animation is yet to revolutionize all industries largely due to costs, practicality and dependence on trained artists. This is an important point as driv-

Figure 1: Example of Performance-driven Facial Capture using markers, used in The Polar Express [1].

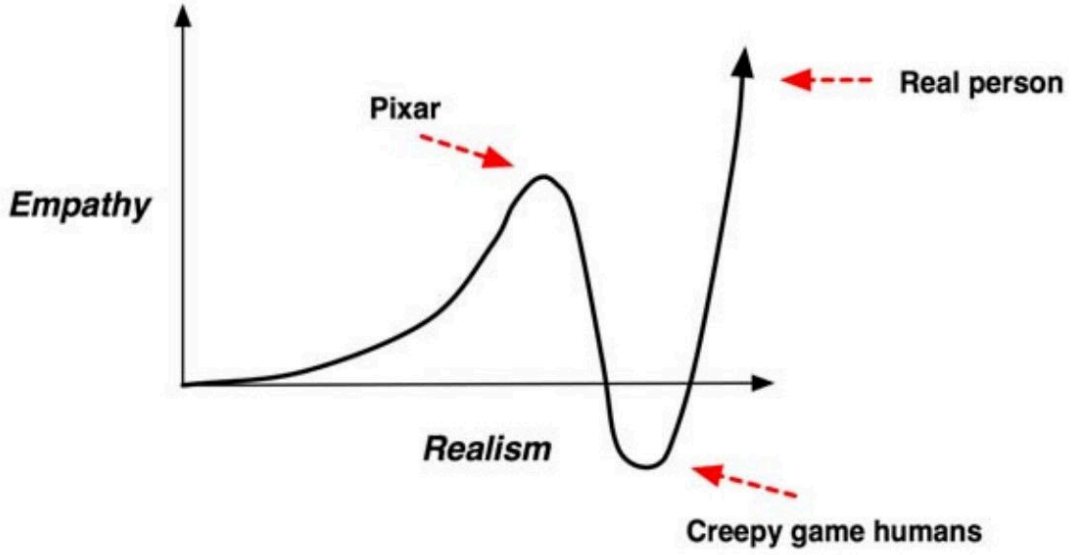


Figure 2: The Uncanny Valley [2]

ing realism isn't the only goal of facial animation. As put by [22], the ultimate goal of this research is a system that creates realistic animation, operates in real-time, is as automated as possible and adapts easily to individual faces. [3] presented Digital Emily, a project that does an excellent job of crossing the uncanny valley into believable animation but still lacks in the other points mentioned i.e being real-time, being independent of the actor and being as automated as possible.

Also, a lot of equipment used for facial capture and animation can be expensive and difficult to obtain for an average consumer. Marker-based systems, multi-camera capture setups

Figure 3: A rendering of a photorealistic digital face from the Digital Emily project, 2010 [3].

and intrusive scanners other than being expensive, usually also require a complex calibration setup. Most systems usually require expertise in setting up and operating them and costs of approximately US$80,000 are required to afford such systems, in addition to a dedicated special recording studio for the project [23]. Owing to this, these approaches are usually unusable at the consumer level. Fortunately, recent developments in this area are moving towards cheaper and easier alternatives that promise the availability of this technology even for the untrained average consumer. With so many factors affecting it, it is clear why achieving convincing facial animation is a very challenging problem.

While the subject is debated, many people feel that artist driven manual key-frame animations may never capture the subtleties of a human face. A slightly different view is that while some skilled animators may be able to produce convincing facial animations, the consistent production of large scale and flawless animations is not practical and very expensive. Owing to this, the trend in facial animation has moved towards using the human face itself as the driver and input device for facial animation — the core idea being that extracting information from an actual performance of facial movements is significantly easier, faster, natural and more scalable than adjusting dozens of sliders. This lead to the idea of *Performance-driven facial animation*. In this context, a 'performance' is understood to be a visual capture of an actor's face talking and emoting which is used to extract information which is used to re-target the motion onto a digital character.

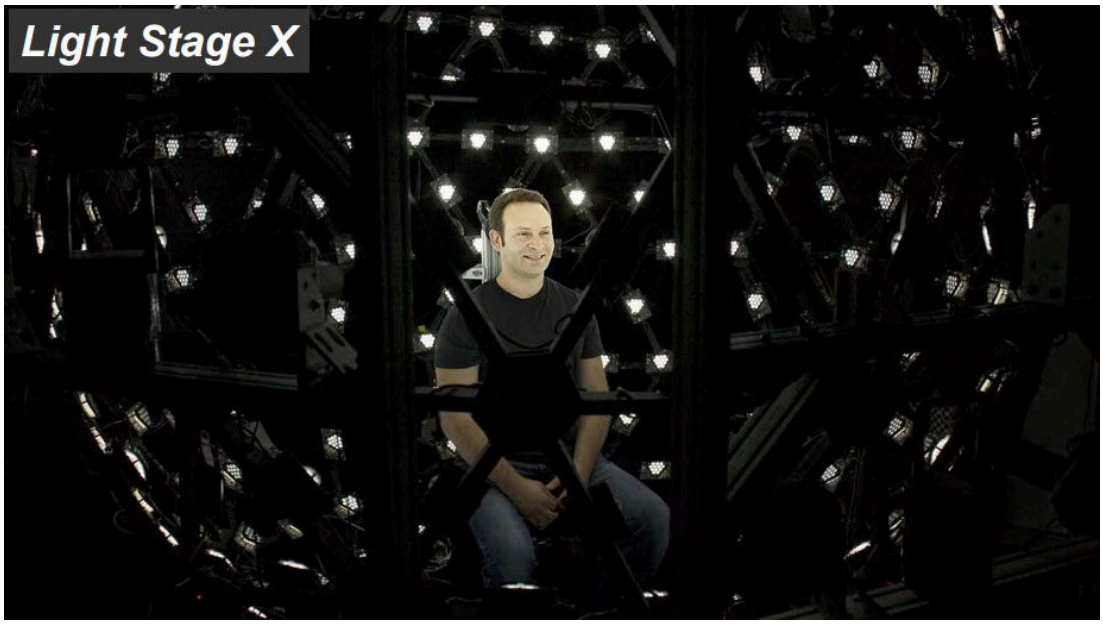


Figure 4: The Lightstage apparatus for scanning faces resulting in extremely high-resolution data [4].

[24] introduced the term Performance Driven Facial Animation to the computer graphics community in Siggraph 1990. Since then there have been numerous works that have extended the original idea. In Siggraph 1992, SimGraphics was demonstrated, which showed a custom face-tracking helmet with physical sensors that tracked gross regions of the face. Hardware motion capture systems were commonplace in the mid 90s and were used regularly in short demos [2].

1.1 Overview

Broadly speaking, the process of Performance Driven Facial Animation can be split into three distinctive stages — Modeling, Capture and Retargeting.

MODELING

The Modeling stage of the pipeline has to do with the underlying representation or 'model' of the human face such that it can be digitally stored, displayed and modified. Over the course of the research in facial animation, spanning decades, multiple methods have been presented, each with their pros and cons. While a clear delineation can be made between the representation, the capture and the retargeting steps, the choice of representation does have an impact on the final animation as the model inherently facilitates or limits the expressive capabilities of the face. Modeling methods range from mesh propagation based methods where a single 3D mesh is

deformed over the performance [5, 8, 25, 26, 27, 28], 2D statistical models [29], 3D statistical models based on PCA [30, 31, 32], Blendshape models [33, 34, 35, 36, 37, 38, 39, 17, 40] to Muscle Based Anatomical models combined with deformable skin [41, 42, 6, 43, 44, 45, 46, 47]. In recent times, mesh based propagation and Blendshape models have been very popular owing to their relative ease of use and intuitiveness. In our work, we make use of a Blendshape model representation for our faces.

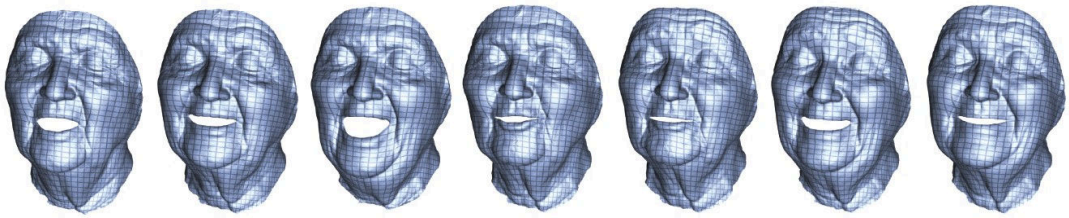


Figure 5: An example of Mesh Propagation being used for the underlying representation for the animation [5].

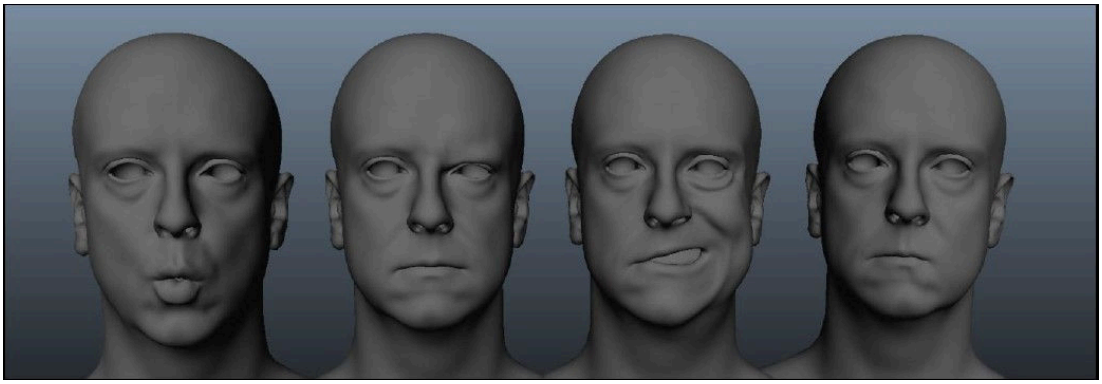


Figure 6: Blendshapes from our Blendshape model for facial animation.

CAPTURE

The tracking or capture stage of the Performance Driven facial animation pipeline can be thought of as the extraction of relevant useful information from the input video or depth information such that this information can then be applied onto the underlying face representation in order to generate the animation. This capture can be done using methods that are active and intrusive or using methods that are passive. Active methods include Marker-based capture where physical markers or dots are placed on the actor's face and tracked through the performance [24, 25, 48, 39, 49, 50, 51] or Structured Light approaches, where a known light pattern is projected onto the actor's face [52, 53, 54, 55]. Passive approaches include methods that use a single or multiple video inputs of the actor's face without any physical markers added on the actor's face [8, 5, 33, 35, 36, 56, 40, 30]. The output of the tracking stage is in the form of

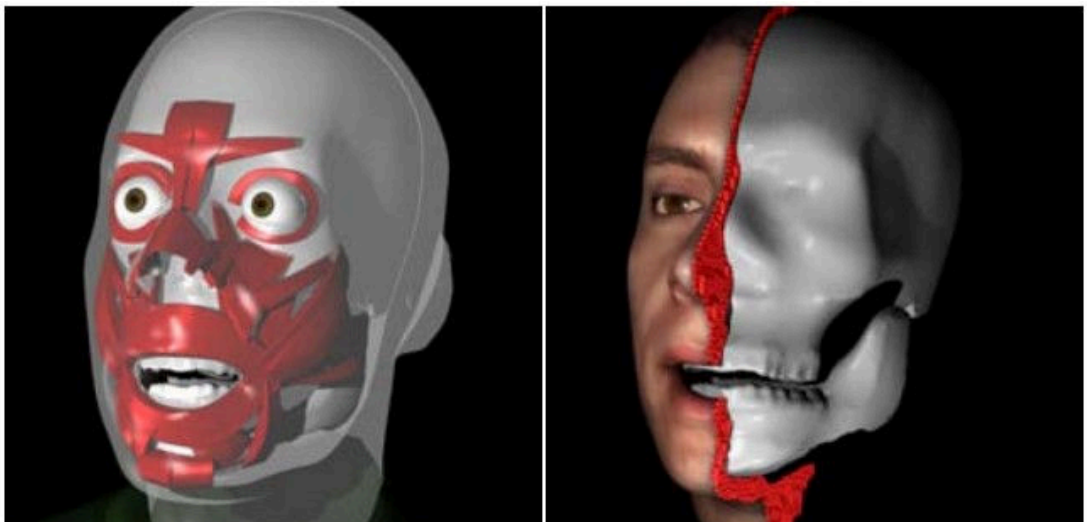


Figure 7: An example of an anatomical model based on muscle activations and deformable skin [6].

parameters of the underlying model that captures, as closely as possible, the performance of the actor, such that it can be recreated with maximum fidelity.

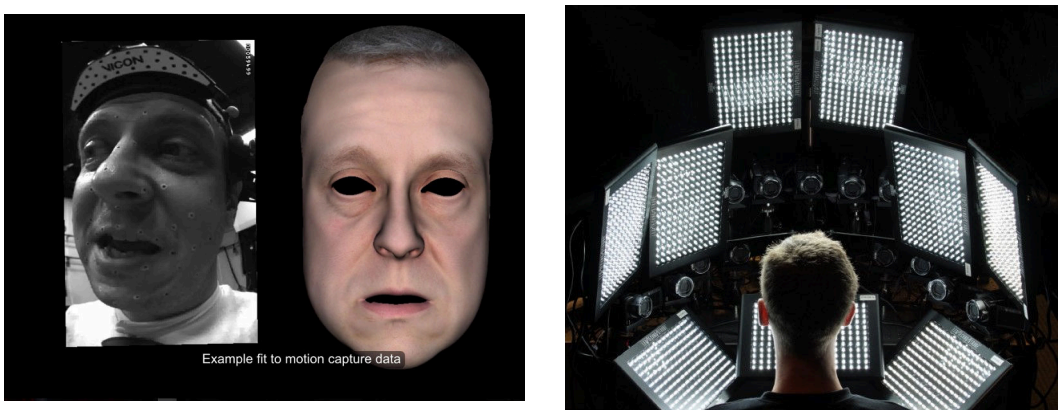


Figure 8: (a) Marker-based capture using a head mounted device [7]. (b) Passive multiview stereo capture setup [8]

RETARGETING

The goal of the Retargeting step is to adapt the parameters obtained from the capture stage and animating the virtual target character. The parameters used to drive this target character can be different from the obtained capture parameters. This is a highly non-trivial task especially when the target character isn't a close replica of the actor and has proportions different from the actor's face. There are many approaches to Retargeting including the use of Radial Basis

mapping [38, 57], Expression Cloning [58], semantic approaches [9, 59], Deformation Transfer approaches [16] and Neural Networks for Retargeting [60].

Figure 9: Retargeting an actor's facial expression onto multiple target characters using parameter parallel layers [9].

1.2 Publications resulting from the PhD

The following publications related to this work were produced during my PhD research:

[61] **S. Ravikumar**, C. Davidson, D. Kit, N. Campbell, L. Benedetti, and D. Cosker, "Reading between the dots: Combining 3d markers and facs classification for high-quality blendshape facial animation.," in *Graphics Interface*, pp. 143–151, 2016

[62] J. Serra, O. Cetinaslan, **S. Ravikumar**., V. Orvalho, and D. Cosker, "Easy generation of facial animation using motion graphs," *Computer Graphics Forum*, pp. 1467–8659, 2017

In addition, portions of the work described in this thesis were included in the following non-refereed materials:

S. Ravikumar, "Lightweight Markerless Monocular Face Capture with 3D Spatial Priors", (Under Review)

K. Reed, **S. Ravikumar**, D. Cosker, Towards the Generation of Personalized Facial Expressions using 3D Morphable Models, *Young Researchers Colloquium, Bristol Vision Institute, UK*, 2016

R. Botham, **S. Ravikumar**, D. Cosker, Extracting 3D Models from 2D Photographs using 3D Morphable Models, *Young Researchers Colloquium, Bristol Vision Institute, UK*, 2017

1.3 Organization of the Thesis

The work in this thesis is broadly split into two main chapters based on our contributions in two significant aspects of Performance Driven Facial Animation — our contributions in the Modeling phase of the pipeline and our contributions in the Capture phase.

Section 1 presents a general overview and introduction to the field of Facial Animation and specifically Performance Driven Facial Animation and outlines the goals of the research in this field along with the open challenges present in achieving these results. It also gives an overview of the main components or stages involved in the process of Performance Driven Facial Animation and the traditional approaches for addressing each of these components. We also present publications resulting from this thesis in Section 1.2.

Section 2 presents a comprehensive and concise analysis of the literature done in the area of Performance Driven Facial Animation over the last three decades. It lists out the various approaches that have been taken in order to address the various challenges present in the Modeling and Capture stages of the animation pipeline and also discusses the pros and cons of each approach. A broad categorization of the numerous approaches to Facial Animation based on the Capture approach is presented.

Section 3 presents our pipeline for automatically generating a complete personalized and editable Blendshape rig with actor specific nuances and ability to easily visualize and debug results generated from our animation pipeline. Section 3.2 outlines in detail our approach for generating a neutral face mesh of the actor starting from a single scan or alternatively, images of the actor. This is the first step in our automated Blendshape generation pipeline. Section 3.3 then presents our automated pipeline that generates actor specific Blendshapes with nuances unique to the actor. Finally Section 3.4 discusses our method to visualize and debug the results of our animation using the rig that was generated using our pipeline.

Section 4 presents our approach and our contributions to the Capture stage of the Performance Driven Facial Animation pipeline. 4.1 presents a detailed description of our marker-based pipeline and discusses our approach for improving traditional marker-based capture by augmenting markers with additional texture patterns and improving the solve results using FACS classification from video. 4.2 presents our markerless capture pipeline in detail and discusses our contributions in the form of using 3D spatial and dynamic priors to improve solve results.

Finally Section 5 presents our conclusions and discussion of future work.

1.4 Overview of Contributions

In this section, we discuss the novel contributions originating from the thesis.

Modeling - Representation of Digital Faces (Section 3)
In Section 3, we present our automated Blendshape generation pipeline and discuss various improvements to the basic approach. Our novel contributions in this area are as follows:

- We outline our entire Blendshape generation pipeline starting from a single Neutral face mesh, obtained either using a scanner or using multiple view geometry reconstruction, all the way to an animatable Blendshape Rig in a Maya framework, complete with individual blendshape controls.
- We present our automatic landmark detection algorithm (Section 3.2.2) for detecting significant points on the face geometry using a HOG [63] based classification scheme followed by a lip/eye contour tracing approach based on Iterative Conditional Modes. We detect these landmarks in UV-space.
- In Section 3.3.4, we present our model for automatic generation of facial expression Blendshapes, by learning a statistical model of Neutral face to Expression Space. We compare our results with other approaches to generate Blendshapes from Neutral expression meshes.

Marker Based Capture (Section 4.1)
In Section 4.1, we present our marker-based capture pipeline. Our novel contributions are as listed below:

- We propose a novel hybrid blendshape optimization (*solve*) which combines two modalities of data: traditional 3D marker data and local facial expression classification based on FACS [64] from video by utilizing the deformation of the sparse make-up patterns between the markers.
- Both sets of information are integrated directly into our optimizer. This allows for improved flexibility by letting just the markers drive the animation when needed and have the classification influence the result when required, thus resulting in smooth and high quality blendshape animations.
- Our classifier is automated and we are able to detect different intensities of Action Units.
- The classifier can be trained once and used on multiple performers.

- Increasing the number of markers on the face introduces further issues in tracking. As we use a texture based classifier trained for specific facial expressions our method is able to handle these situations.

Markerless Capture (Section 4.2)
In Section 4.2, we present our markerless capture pipeline based on learned 3D spatial priors. Our novel contributions are as follows:

- We present a lightweight monocular markerless capture method that achieves good quality animation parameters and does not require special equipment, controlled lighting environments or complex training phases.
- We exploit easily available prior animation data obtained from 3D tracking systems and use this in a density estimation framework to regularize our objective function and generate more plausible results. We enable further flexibility by learning separate prior constraints for the upper and lower face regions.
- We combine initial estimates of 2D landmark points on the face based on an ensemble of regression trees, with an Active Appearance Model for improved accuracy.
- We handle noise in input 2D features and in the estimation of camera extrinsic parameters thus eliminating jitter in the resulting animation.
- We also present our lighting optimization approach that uses the inverse rendering equation to find the optimal values of light intensity and position thus enabling us to relight, re-texture and overlay the mesh on the original video.

2 Background

2.1 History

Facial animation can be traced back all the way to the pioneering work of Parke et al. [18] in 1972, who outlined a method of representing and animating human faces digitally. They describe a polygonal model for a face created using photogrammetry, controlled by shape interpolation. Parke et al., 1974 [65], followed this by developing the first parametrically controlled face model. This work laid the foundation for a lot of the work in facial animation that we see today. The early 1980's saw some work on anatomically inspired muscle-controlled face models by Badler et al. [66] who built a face model using masses and springs including forces generated by muscles, and made use of the Facial Action Coding System (FACS) [67] for measuring and indexing facial behaviours. Waters et al. [68] in 1987, describe a more generalized muscle model. Around the same time, Lewis et al. [69] used speech driven animation which mapped a set of phonemes to mouth shapes of a parametric model. This was extended by Cohen et al. [70] in 1993 to include coarticulation.

Williams [24], in 1990, introduced the method of 'Performance Driven Facial Animation'. Since then countless works have built up on this method and presented multiple approaches to both representing and animating human faces. The progress made in the realistic rendering and animation of faces has been tremendous over the last few decades and near convincing results, arguably indistinguishable from real faces in both look and motion, have been achieved [4, 71, 3]. Numerous different methods for tracking, modeling and retargeting have been proposed over the course of these decades, each with their pros and cons. In this section, we cover these different approaches and point out the strengths and weaknesses of these approaches.

2.2 Related Work

As mentioned previously, the process of performance driven facial animation can be delineated into three broad stages, Capture, Modeling and Retargeting. The tracking or capture stage of the performance driven facial animation pipeline deals with the acquisition of the data either in the form of sparse or dense information which effectively 'captures' and parameterizes information about the movement of the actor's face, including facial expressions, lip-movements and other subtle facial movements like wrinkles, bulges and twitches. The form that this acquired data will ultimately take depends on the underlying representation that is specified by the choice of Modeling approach, but the actual process of acquiring this data falls under the umbrella of tracking. Widely varying methods for accurately tracking the face have been proposed

over the years and while these can be reasonably categorized based on a few core distinguishing traits in the approach, in practice there is a lot of overlap between methods and many methods use a combination of these varying approaches. Nevertheless, these works can broadly be distinguished based on whether they are — sparse vs dense methods, active vs passive methods, marker-based vs markerless approaches, real-time vs off-line approaches, monocular vs multi-view methods and methods that use depth-sensing devices vs ones that use only 2D inputs. To add to the complexity, a lot of these methods that share capture pipelines, might differ in their choice of Modeling approaches. Modeling methods can broadly be categorized as — Mesh Propagation approaches, statistical models based on PCA, Blendshape models and physically inspired anatomical models that combine muscle activations with a polygonal skin representation. Given that a clear taxonomy of methods is not straight-forward owing to there being significant overlap, in the following sections we categorize the methods based on capture techniques and point out when they differ and where they are similar both in their choice of tracking and the modeling approaches.

2.2.1 Marker Based Methods

Williams [24], 1990, first introduced the idea of tracking physical markers placed on the face in order to capture the performance of an actor. They obtain a single face mesh by scanning a plaster cast of the actor's face using the system of Cyberware, Inc. [72]. They stick retro-reflective markers on the actor's face, placed manually to avoid proximity to each other, and track the movements of the face from a 2D frontal video input. They use a mesh deformation approach where the single mesh of the face is deformed over the sequence using a set of warping kernels distributed about the face. They make use of a beam splitting apparatus in order to improve the reflection of light from the markers onto the camera, and assume that there is little head movement and the view is always frontal. While this work pioneered the area of Performance Driven Facial Animation, the results are very basic and the method is not robust enough for practical use.

Guenter et al. [25], in 1998, presented a complete marker-based pipeline for the acquisition and animation of a face by using multiple camera views of the face and reconstructing the marker locations in 3D. They scan and digitize the actor's face and use a mesh propagation approach in order to deform the face through the animation sequence. They cover the actor's face using 182 fluorescent markers of different colors, placed along the contours of the face and track the movement of these markers using 6 synchronized and calibrated cameras. The actor's face is illuminated using a combination of visible and UV light in order to make the markers stand out. They make use of a color classifier to detect pixels in the video texture and

track and label the centers of these markers throughout the video sequence and reconstruct the 3D positions of the markers. In order to animate the 3D mesh of the actor's face, they use a 2 step process where they first use the marker positions to interpolate a set of evenly spaced grid-points on the 3D face based on their neighboring markers and follow it up with a second step where they deform the vertices of the mesh based on their neighboring grid-points. In a post-process step, they obtain the final face texture by removing the markers from the video texture from each of the 6 cameras by substituting the colored pixels with skin texture. While their method provides good quality animations, they require studio settings, multiple cameras, manually placed markers and a scanner in order to obtain the 3D mesh. Their method is not real-time and requires complex post-processing for obtaining good texture information. This makes it difficult to apply on a consumer level without costly equipment and expertise.

Choe et al. [46] attempted to bridge the gap between performance driven facial animation and physically based modeling. They detailed an anatomical model of the face represented by muscle activations and a skin surface. Their model consisted of 19 parallel and three sphincteral muscles,and a rotate-able jaw. The skin surface deformation was calculated using a finite element method. They drive this model using motion capture data obtained by reconstructing twenty-four 3D markers on the actor's face from three synchronized and calibrated digital video cameras. They solve for the muscle activation parameters by solving for a linear model and use these parameters to synthesize expressions by by sending the data to the finite element method. As stated by [46], "An advantage of using muscle actuation parameters is that retargeting of an expression to other faces becomes a trivial job. This is based on the assumption that even though muscle size and layout are different for each individual, people use the same actuation pattern to make a similar expression. Another important advantage is that the muscle parameters can be easily converted to higher-level control parameters such as Facial Action Coding System or MPEG-4 Facial Animation Parameters."

In 2005, Sifakis et al. [6] developed an anatomically accurate model of facial musculature, tissue and underlying skeletal structure using volumetric data obtained from a living male subject. This included a triangulated surface for each bone, a tetrahedralized volume and a B-spline fibre field representation for each muscle and a single tetrahedral mesh for all the soft tissue. This face model was driven by motion capture markers and controlled by muscle activations and kinematic bone degrees of freedom. They were able to obtain visually plausible and anatomically accurate deformations and their system was robust to outliers in motion data. Furthermore, they show that their system can interact with external stimuli in a realistic manner. Their template head model was created by a graduate student over a period of 6 months from visible human data and this template is then morphed to fit data obtained from both laser and MRI data to create a subject specific model. Their model creation and morphing process is

quite work intensive and requires some expertise. It also includes eyes and teeth. The important aspect of their approach is that the search for the optimal control parameters to fit the motion capture data is done over the space of physically plausible configurations, parameterized by muscle activations. The flesh is driven based on the muscle activations using a finite element method. As their model is anatomically accurate, the obtained activations have a biomechanical meaning to them making it useful in medical applications.

Borshukov et al. [48] in 2006, presented a facial capture pipeline that is very similar to the work by [25] but they extend this work to include compression of the data which allows them to achieve interactive rates of animation in the compressed space. Their system uses 8 IR cameras plus 3 synchronized, high-definition, color cameras framed on the actors face as well as an ambient lighting setup. 70 small retro-reflective markers are placed on the actors face. The IR cameras track the light reflected from the markers and simultaneously the hi-def color cameras capture the subtleties of the actor's performance. The two data sources are aligned in time and in space. They make use of a facial bone rig, instead of direct mesh propagation like [25], and the rig is driven by the marker motion. The number of bones is equal to the number of markers and the face is moved by using standard skinning approaches. They additionally address lip-movement separately by assigning separate bones attached to the inner contour of the lips which are manually adjusted by animators using the video as reference. They obtain the texture in a fashion similar to [25] et al. by projecting the mesh into the different camera views and then removing the markers by replacing them with skin texture. Their most important contribution is that they highly compress the large data sets allowing them to achieve real-time interactive rates. They propose a novel compression method based on a novel variant of the PCA compression algorithm that varies the number of components used to represent data. They go on to show the application of this compression by constructing a motion-graph for faces and allowing for linear interpolation between poses at interactive rates.

Bickel et al. [51], in 2007, presented a complete marker-based pipeline that addresses multi-scale representation and acquisition of facial geometry including wrinkles on the face. Their method decomposes the facial features into fine, medium and coarse spatial scales, each representing a different level of motion detail. They first acquire a static scan of the face using a combination of a commercial scanner combined with photometric stereo. They also capture high quality reflectance data including the texture albedo, spatially varying BRDF, and sub-scattering parameters. A traditional motion-capture system using 6 synchronized cameras at 50 fps for triangulating marker locations in 3D is used and on top of this 2 synchronized high-resolution cameras for tracking the wrinkles are added. They place 80 to 90 blue markers on the face and mark expression wrinkles with a diffuse color. All the cameras are extrinsically calibrated. The large scale motion is obtained from the marker data by deforming the high-

resolution mesh using a linear shell based deformation. They use an image-based algorithm for tracking wrinkles in video data, fitting 2D B-splines to the wrinkle valleys, and estimating their cross-section shapes from self-shadowing effects. They then use a physically-inspired nonlinear shell deformation model that, with the 2D data as input to synthesize medium-scale 3D expression wrinkles and bulging onto the large-scale animation. Like the previous works before them, they assume uniform ambient lighting and that the subject faces the same direction through the sequence, with little movement. While this model is suitable for performance capture and replay it does not provide intuitive parameters for animation control as the motion is in the form of vertex displacements on the mesh.

Bickel et al. [73], 2008, extend the previous work to obtain a pipeline for real-time animation of highly-detailed facial expressions including proper parameterization of wrinkles so that they can be efficiently stored and transferred to other face models that do not share the same topology of the original mesh. Similar to [51], they calculate the large-scale motion using a marker-based capture system, but they augment this with what they call 'Pose Space Deformation' which learns the correspondences between sparse measurements of skin strain to wrinkle formations from a small set of example poses. Given the example poses, the corresponding fine-scale details are extracted as the difference between the examples and the results of the large-scale deformation for the same poses, and are stored per-vertex in local tangent frames. They construct a feature graph who's edges constitute the regions where skin strain is measured. For each new pose in the animation, they compute the skin strain according to the deformed feature graph and use this information to generate wrinkles. Their pose space deformation is learn as a scattered data interpolation problem using radial basis functions. The transfer of wrinkles to other face models is done by first establishing correspondences between the models using a non-rigid deformation step after which the corresponding feature graph for the new face can be calculated in a straight forward manner.

Ma et al. [74] in 2008, present a method of Polynomial Displacement Maps, quite similar in spirit to [73], but their method captures not only wrinkles but also dynamic fine-scale pore detail. The method consists of an analysis phase where the relationship between motion capture markers and detailed facial geometry is inferred, and a synthesis phase where novel detailed animated facial geometry is driven solely by motion capture markers. They use a combination of structured light and photometric stereo to obtain high-resolution face scans and a stereo pair of high-resolution high-speed cameras synchronized to a video projector and a spherical gradient illumination device similar to that of [75]. They place 178 tracking dots on the actors face so that each frame of motion can be registered in a common texture space and also serve as the basis for the parameter space for facial detail synthesis. Several short sequences as the subject transitions from the neutral expression to various strong expressions are captured

and they select between 10 and 30 frames to use as input to the fitting process. They derive the base mesh using a linear thin shell interpolation technique to deform a neutral mesh to the basic shape of the current expression, as in [51]. However, instead of using the relative length of each edge in the feature graph as in [73], they approximate 2D strain as the difference between the standard deviation of the projected positions of vertices in the current deformation from the reference neutral expression. Given new poses, they are able to first solve for the base mesh shape and synthesize both medium scale wrinkles and fine-scale pore details.

Huang et al. [76] describe a face capture pipeline using a marker based system that allows users to automatically select a minimal set of base poses from a performance that can then be used in a Blendshape representation. Their method captures fine-scale details including wrinkles and pore-level details and matches both the spatial resolution of static face scans and the acquisition speed of motion capture systems. Their pipeline uses the Vicon [7] marker based system using 12 cameras and 100 markers on the face, combined with an RGB camera for reference to capture a performance of an actor. The selection of the minimal set of poses from this sequence is posed as an energy minimization problem. The approach is essentially a greedy algorithm that adds poses to the basis set one by one if it reduces the overall reconstruction error and keeps adding poses until a threshold is reached. Once the set of poses has been selected, they have the actor perform those specific poses again and scan them using a laser scanner in order to obtain high-resolution meshes including pores and wrinkles. They then register the markers for those poses with the scans in a 2-step energy minimization framework where they first solve for the rigid transformation between scanned pose and markers, and then solve for the non-rigid transform in the form of a weighted sum of the selected marker poses and perform these steps iteratively until convergence. Finally all the scans are put into dense correspondence using a 2-step process involving a Laplacian Deformation technique for large-scale deformation, followed by a fine-scale deformation for registering wrinkles and pores. This fine level registration is done by splitting the face into 8 separate regions and projecting the 3D vertices into a cylindrical space to obtain images and then performing optical flow in image space for each of the regions with their closest neighbors. The offsets are then projected back to 3D in order to obtain the fine-scale registration between the poses. These high-resolution base poses are then utilized as Blendshapes to solve for the entire performance.

Matthews et al. [31], in 2011, presented a region-based model that is composed of a set of PCA sub-models which are independently trained, but share boundaries. They show that the region based model generalizes better than its holistic counterpart when describing motion capture data. The sub regions allow us to interactively modify the model at a local level. The method solves for all the sub-regions simultaneously and enforces boundary consistency in a soft least squares sense which allows discrepancies at the inter-model boundaries. The decomposition

of the face into regions is determined automatically from training data, the only requirement being that each region shares at least one vertex with at least one other region. They achieve this automatic segmentation by grouping vertices on the face that are highly correlated and ones that are physically close to each other. In order to do this, they use the data from a face mesh pre-fitted to a motion capture data consisting of FACS sequences, emotional speeches and range of motions. They calculate a normalized correlation matrix of all the vertices on the mesh separately for the x,y and z axes and average it into a single correlation matrix. They also calculate the inter-vertex distance on the mesh using an isomap algorithm to form another matrix. These 2 values are combined using a weighting factor resulting in an affinity matrix. Finally spectral analysis using k-means algorithm is performed on the affinity matrix resulting in segmentation of vertices. The weighting factor controls the number and size of the segments on the face. The main limitation of the method is that the expressiveness of the model is limited by the training data.

Bhat et al. [39] present an artist-friendly performance capture pipeline that focuses on the perceptually important contour features on the face. They augment a traditional marker-based system with information obtained from the texture in order to improve their solve results. They incorporate the silhouette contours of the eyelids and the inner mouth to reconstruct accurate animation. They use an effective heuristic to dynamically associate the tracked curves with edge contours on the mesh at every frame. The system uses two synchronized HD cameras attached to a helmet worn by the actor. These cameras are static with respect to the actor's face. Markers are placed on the actor's face and in addition, they manually trace the outline of the upper and lower eyelids as well as the silhouettes of the inner lip. In order to match the curves with the face vertices, they choose an edge contour that has the maximum number of silhouette edges, where a silhouette edge is defined as an edge shared by a visible and invisible polygon. They then find correspondences between contour vertices and the curve by projecting the contour vertices into the image, aligning the end points of the contour and the curves and then use an arc-length based mapping. Their method also makes use of an additional prior which takes into consideration the fact that eyelids slide on the surface of the cornea. They incorporate this prior by projecting the eyelid curves onto the cornea from the camera to obtain a 3D curve which they also use within their 2 step optimization. A standard Blendshape solve is used to obtain the basic shape and finally, in order to improve the fit, an out-of-subspace corrective is performed in the form of an energy minimization that uses a cotangent weighted Laplacian constraint for regularization. This enables them to accurately capture shapes that are not achievable using the Blendshape model itself.

2.2.2 Image Based Methods

Yacoob et al. [77] in 1995, proposed the use of local parameterized models of image motion that can track both rigid and non-rigid facial motions. They interpret these parameters to recover high level semantic interpretations of facial motions, including recognizing the six universal facial expressions (surprise, sadness, anger, happiness, disgust and fear). Their core assumption is that parameterized models of image motion within a region can be represented by a low-order polynomial. Within small regions on the face, they use an affine model to detect translations, divergence, curls and deformation of facial features. They further augment this model with parameters for yaw and pitch in order to better detect rigid planar motion of the face in the video. Non-rigid motions such as curvature of the eye-brows and mouth are captured with the addition of another parameter to the affine model. The curvature parameter is only able to capture very coarse curvatures and cannot deal with asymmetry but is able to capture the essential motions for recognition of the basic facial expressions. They recover these parameters from the image using a robust regression approach. The parameters that are obtained in this way are then used to derive mid-level facial motions such as horizontal or vertical movement of the mouth or eyebrows, rotations of the head and also high-level expression recognition. The obtained facial expressions can be divided into temporal segments including beginning, apex and end.

Essa et al. [78] drive an anatomical muscle based model, combined with a polygonal skin mesh using a monocular video input. Their system uses optical flow measurements of surface motion as input. They automatically detect features around the eyes, nose and lips in the image and use these to register their face image with the canonical face mesh. This enables them to extract additional feature points on the image that correspond to fixed nodes on the face mesh. After this initial registration, optical flow is used to obtain pixel-by-pixel motion estimation that tracks the movement of the head and the face, as long as there isn't excessive head motion. The motion vectors obtained from this are projected onto the mesh in order to obtain the deformation of the skin from which the muscle activations required to produce that deformation is estimated. In their work, they make an assumption that the input video contains a limited set of facial motions.

Terzopoulos et al. [79] proposed the concept of a dynamic, elastically deformable model for inferring the structure and motion of non-rigidly moving objects from images. The proposed model has intrinsic constraints that enforce certain properties on it, such as surface coherence and symmetry around an axis with room for deviations, and is subject to extrinsic forces which are inferred from the images. They show that this model can be extended to temporally varying non-rigid objects and to multiple views. [80] in 1996, propose a method for the integration

of the deformable model with optical flow as an external constraint and drive a face model. Their deformable model is a polygonal face mesh with 10 component parts and the shape and the motion of the face model is specified by parameters. The variations in the face are specified using a number of deformations — translations, rotations, scaling and bending. Each deformation on the face is specified by a small number of parameters and is applied locally to parts of the face ranging from a single part to the whole face. This model is produced by a designer who carefully combines these deformations. Their system uses optical flow and forces computed from edges simultaneously. The tracking of the edges is facilitated by prior knowledge of what locations on the face model are likely to produce edges in the image. They only make use of a subset of image points to calculate the optical flow. As they make use of optical flow, they make an assumption of photometric invariance to satisfy the optical flow constraint.

Most of the previously mentioned image based approaches are able to detect coarse expressions and coarse movements but do not achieve the quality required for realistic facial animation. The motion is restricted to coarse and general motions and are unable to capture subtle movements and nuances.

Pighin et al. in 1999 [81] present a method to automatically recover the position of the face and the facial expression from each frame of a video sequence. Their method fits a linear combination of 3D texture-mapped models to the video, each one corresponding to a basic facial expression — joy, anger, sadness, surprise, disgust, pain. The face is parameterzed using 2 subsets of parameters, one subset for the translation and rotation parameters and one corresponding to the coefficient weights for the linear combinations. They constrain the sum of these coefficient weights to sum to 1 using what they call 'expression parameters' and provide a mapping from the expression parameters to the weight parameters. In order to span a wider range of facial expressions, the face is split into several regions that are controlled independently, each with separate expression parameters. Each of the basic expressions are rendered and these renderings are blended together using the weights to produce the final image. The goal of the optimization is to find parameters yielding a rendering that best resembles the target image. Finally they discuss the applications of their methods for relighting, changing perspective and adding textures on top of the face in the video.

Cootes et al. [29] in 2001, described a method for matching a statistical model of faces to images. Their statistical appearance models are generated by combining a model of shape variation with a model of texture variation. The training data consists of face images marked with points defining the main features which are aligned across images and a statistical model of shape is built. The training images are then warped to the mean shape to obtain a shape-free patch which is used to obtain the texture vectors which are then normalized to obtain the

texture model. The correlations between shape and texture are learned to generate a combined appearance model. Their method uses a multi-resolution model based on a Gaussian image pyramid which fits the model from coarse to finer resolutions allowing for improved speed and robustness. Finally, they present an iterative approach for matching the statistical model to a new image. This method makes use of learned correlations between errors in the model parameters and the resulting residual texture errors and converges very rapidly and reliably when given a reasonable initial starting position.

Bregler et al., 2002, [82] present a Blendshape based method that decouples the capture and retargeting phase so that only the Blendshape weights need to be transferred from source to target. Their method automatically tracks facial features from a source video sequence and extracts key shapes from the tracked data. Their feature tracker is trained on an annotated database labelled with facial contours. They present an automatic method to choose appropriate key shapes which they use within a Blendshape framework to represent the entire animation sequence. Their method chooses the optimum number of shapes to be used based on multiple heuristics based on maximum spread along principle components, clustering in low dimensional space and choosing shapes that lie along a convex hull in low dimensional space. They evaluate each of these methods. For the retargeting of the recovered facial motion to the target shape, the user creates new key shapes for the target model which resemble the key shapes for the source animation and the decomposed weights for the key shapes in the source video sequence are used to interpolate the target facial expression.

Chai et al. [83] in 2003, showed that preprocessed motion capture data can be used to generate rich life-like facial actions and that the user can control these actions by using a single video camera. They go on to present a retargeting technique that is independent of the complexity of the character model. Essentially, their method uses knowledge embedded in the motion capture data in order to convert the low quality and noisy control signals into high quality facial animations. The motion capture data for the pre-processing is obtained using a Vicon system and 76 reflective markers, combined with a laser scanner for obtaining the surface model of the capture subject. Each motion capture frame is associated with animation control parameters obtained from the data. During capture, control paramters are obtained automatically from features detected from the video. The system extracts 15 scalar quantities describing the movements of the mouth, nose, eyes, eyebrows based on the positions of the features which are used to map to the high-quality motion data. The motion synthesis consists of a normalization step that corrects for differences between the control parameters in the tracking and the motion capture data, followed by a data-driven filtering of the noisy data and a data-driven synthesis approach to convert the filtered data into high quality motion. The key idea of the filtering technique is that they use a low-dimensional linear subspace to approximate the local region of

the high-dimensional nonlinear manifold. Their facial tracking system runs in real time at 20 fps and is user-independent.

Vlasic et al. [84] in 2005 demostrated a multilinear model of 3D face meshes that separately parametrizes the space of geometric variations occuring due to identity, expression and visemes. These parameters are used to drive a 3D textured face mesh which can be seamlessly rendered back into the target footage. Their method also allows for editing the performance in order to change identity, expression and pose independently. Their multilinear model is constructed by putting a range of 3D face scans performing multiple facial expressions including neutral, smile, frown, surprise, disgust and others. They also present a method of imputation for filling in missing data in the tensor using a PPCA [85] formulation. The system is able to automatically set model parameters from video data using optical flow in conjunction with a weak-perspective camera model and obtains the texture from the video itself. Their method is able to produce video-realistic results for new source and target subjects even with a model estimated from small datasets.

Cootes et al. [86] in 2006 presented the Constrained Local Models approach, that extends the method of [29] to generate likely feature templates iteratively instead of approximating image pixels directly. Their method is more robust and accurate than the original AAM search method. The system learns the variation in appearance of a set of template regions surrounding individual features. Their joint shape and texture model is built from a training set of 1052 manually labelled faces. Given a new input and current image points, the template generation process fits a joint model of shape and appearance to regions sampled around each feature point. These templates are then used to search using Normalized Cross Correlation, generating a set of response surfaces. The parameters of the shape model are then optimized to maximise the sum of responses at each point and the search then proceeds iteratively.

Chai et al. [87] presented an interactive system that allows any naive user to model realistic facial expressions quickly and easily. Their system learns facial priors from pre-recorded facial expression data which are used to generate natural facial expressions that match the user's constraints. The facial modeling problem is formulated in a probabilistic framework by combining the user's constraints with the facial priors. Their pre-recorded data is obtained using a Vicon system using 55 reflective markers and consists of the subject performing a wide variety of facial actions including the basic facial expressions and speaking motions. They convert these recorded motions into a set of deforming models and use PCA to learn a reduced subspace representation of the data in an offline step. The user's constraints can be specified iteratively as either point, stroke, distance and curve constraints which can be specified in 2D screen space until desired results are obtained. At run-time, the system solves for the maximum a posteriori solution that satisfies the user's constraints and the statistical properties of the captured

data. They go on show the application of their method for facial expression transfer between different subjects.

Furukawa et al. [88] in 2008, proposed an approach for non-rigid and markerless motion capture from synchronized video streams acquired from calibrated cameras. Their method represents the geometry of the observed scene using a polyhedral mesh of fixed topology, which is constructed in the first frame using stereo reconstruction software. The deformation of the face is captured by tracking the vertices of the mesh over time by optimizing for a local rigid motion in the neighborhood of each vertex and a global non-rigid motion for the whole mesh. The tracking algorithm consists of a local motion estimation around each vertex which is decomposed into normal and tangential components and made robust using an expansion strategy based on spatial consistency, a global deformation that is regularized for smoothness and tangential rigidity and finally a filtering step to remove erroneous motion estimates. These three steps are iterated for each frame. Their method is able to handle complex, long range motion and also errors due to occlusions. The method assumes locally rigid motion and is not designed for non-rigid deformations with much stretching, shrinking or shearing which are common in facial expressions. [89] go on to show that the tangential rigidy assumption made in the previous work does not work well with intricate facial expressions and present a solution to this by modeling the tangential non-rigid deformation.

Bradley et al. [8] present a passive multi-camera setup that leverages the pores, blemishes and hair follicles on the actor's face and uses them as trackable features. Their method makes use of an array of cameras and does not require any template facial geometry, makeup, markers or active lighting. The setup consists of 14 calibrated and synchronized HD cameras arranged in 7 stereo pairs and each pair is zoomed on a small patch of the face surface in high detail. They make use of an iterative binocular stereo method to handle outliers and reconstruct each of the 7 patches and then combine them into a single high-resolution mesh. They then propagate a single reference mesh through the entire sequence using optical flow in order to obtain a consistent temporal reconstruction. The drift inherent in the optical flow is detected in the per frame texture maps and corrected. Temporal drift in the 3D geometry appears as a small 2D shift in the texture domain, which is detected by optical flow again. In order to handle mouth movements during high-speed talking motions, they use a sparse set of points around the mouth and impose positional constraints on these in image-space. Finally, they post-process the sequence to provide a smooth realistic facial animation using a saliency based smoothing technique which preserves more salient features while smoothing the less salient ones.

Beeler et al., 2011 [5] make use of 'anchor frames' to handle drift in their optical flow based passive approach. They leverage the fact that facial performances contain repetitive subsequences and they automatically identify frames which contain facial expressions similar to a

reference frame. The system treats each segment between these anchor frames independently allowing for parallelization and also for splicing unrelated clips. They use a set of seven synchronized cameras and uniform illumination. The anchor frames are detected automatically using normalized cross-correlation and the sequence is partitioned into clips. Image pixels from the reference frame are then tracked to the anchor frames first and then sequentially to all the non-anchor frames. The reference mesh is then propagated through the sequence using the optical flow motion field, providing an initial estimate of the face motion. This is refined further in two stages – independently on each frame to optimize for photometric consistency and smoothness and across frames for temporal coherence.

Saragih et al. [90] in 2011, presented a method for real-time facial puppetry. Their system runs in real-time and does not demand any special hardware. The actor's face is tracked using a real-time 3D non-rigid tracking system and expression transfer is achieved by learning a mapping from user expressions to avatar expressions. It consists of an offline phase where a basis of variation that captures changes in shape and texture is learned from an annotated database of images. A mapping from neutral facial shapes to a set of discrete facial expressions is learnt from this same database and is used to generate synthetic facial shapes for both the user and the avatar. In the online phase, the user's face is tracked in order to obtain the shape and texture which are then mapped onto the avatar using the mapping learned in the offline phase. Their system also provides for a failure detection mechanism that allows it to recover from cases where it fails to track the face by using a linear support vector machine to distinguish between aligned and misaligned configurations. Gaze tracking is performed by detecting the pupil.

Valgaerts et al. [56] show that good quality facial capture can be done in uncontrolled and time-varying lighting even in outdoor scenes using just a binocular stereo rig. Their system tracks a coarse face template through the binocular sequence in order to provide a sequence of coarse meshes that are in full correspondence with each other using an image based scene flow method that makes use of brightness constancy assumptions on image pixels across cameras, geometric relations enforced by epipolar constraints between the cameras and smoothness constraints for regularizing the motion. The mesh is further refined by projecting it back into the image and using optical flow to detect corrections which are applied to the mesh. In a second pass, fine-scale time varying details such as wrinkles and folds are added onto this coarsely tracked mesh sequence. The algorithm makes use of shading cues in a two stage process where a clustering approach is used to obtain albedo groups on the face which are used to estimate the albedo values and incident lighting per frame, using which the coarse geometry is refined by displacing each vertex along its normal in a MAP framework. Any remaining flicker in the animation owing to differences in surface normal direction across frames is handled by averaging normals over a temporal window and adapting the geometry accordingly.

Garrido et al. [36] present a lightweight approach that makes use of a single monocular video input and are able to generate a high quality output animation with fine scale details using a Blendshape model. They track a few sparse landmark feature points on the face reliably using forward and backward optical flow combined with automatic key-frame selection based on local binary patterns for robustness. The pose and facial expressions are estimated from these sparsely tracked points on the face in an iterative fashion. Temporally coherent dense motion fields tracked from video combined with a smoothness constraint is then used in order to refine the pose and facial expression. Fine scale details are then added on top using a shape from shading approach.

Beeler et al., 2014 [91] learn a model of expressiveness for an individual that encodes information about subtle spatial and temporal deformation details specific to that individual. This information is learned from a high-resolution facial capture system that is used to acquire a representative performance of that individual in which they explore the full range of facial motion. This information is then used for adding fine-scaled details and expressiveness to a low-resolution capture sequence. The system also leverages the timing information in the database and uses it to enhance facial key-frame interpolation to include non-linearities in motion. They use the method of [5] in order to obtain their high-resolution capture database and then perform a frequency separation step in order to isolate high-frequency components of the performance. They encode this information in a shape-space defined in deformation gradients representing the stretching and rotations of triangles which allows for matching, projection and interpolation. Given a new performance, each frame is first brought into correspondence with the database geometry, encoded into shape-space and projected into the database shape-space in a matching step after which the relevant high-frequency information is interpolated and added onto the original animation.

Shi et al., 2014 [92] present an end-to-end facial capture system that makes use of a multilinear face model and is able to reconstruct the pose, large-scale facial deformation and fine-scale details from uncontrolled monocular videos. This includes a facial feature tracking algorithm based on a per-pixel classification scheme using a random forest classifier that attributes a probability to each pixel of being an important facial feature. Their system then makes use of a preexisting database of labeled images in order to search and obtain closest examples. They combine this with an Active Appearance Model and and facial priors in order to obtain the tracked features. The large-scale motion of the face is obtained by finding the appropriate identity and expression parameters of the multilinear model in a space-time optimization framework regulated by smoothness and expression/identity priors. Finally, the fine-scale details, unknown incident lighting and face albedo are estimated by minimizing the inconsistency between hypothesized and observed images and obtaining per-pixel normal maps. The process

is iterated by refining the large-scale geometry using the per-pixel normal estimates and then updating the normal maps with the large-scale deformation again.

Kemelmacher et al. [93] also present a method that takes an uncontrolled input video and reconstructs high detail 3D shapes for each video frame. They do not aim for geometrical accuracy to ground truth, but rather attempt to obtain convincing reconstructions. Their method leverages the large amounts of photos which are publicly available on the Internet and use this to obtain an average face geometry of the individual and also its appearance under a subspace of illuminations as described in [94]. Their pose estimation is done using an iterative algorithm that uses optical flow to find correspondences between projected mesh and the 2D image and made temporally coherent by averaging pose estimates across neighboring frames. They then use a novel 3D optical flow system that computes dense correspondence between the mesh and the video for every frame and deforms the reference mesh in order to fit the video. High frequency details are added on top of this basic deformation by further deforming the mesh such that the rendered mesh fits the shading in the video as closely as possible.

Cao et al., 2013 [33] presented a method for obtaining real-time performance capture from monocular input. Similar to our approach, they track 2D points but instead of fitting directly to the 2D features, they train a user specific two-level boosted regressor trained on labelled 2D points and corresponding 3D shapes, in order to map from 2D to 3D features at run-time. They then fit a Blendshape model to the obtained 3D features by iteratively solving for transformation parameters and expression weights. Their method requires a training phase where images of the user in different poses and expressions are captured. Cao et al., 2014 [34] extend the previous work to be independent of a user and instead learn a regressor from public image databases. They infer both the 2D facial landmarks and the 3D shape of the face simultaneously. Their algorithm adapts to the user's face at run-time by solving for the user-specific Blendshapes and the expression co-efficients in an iterative manner. Cao et al. 2015 [35] enhance a low-resolution tracked mesh with medium-scale wrinkle details which are generated by local regressors trained on high-resolution scan data. They require a one-time training phase in order to learn the mapping from UV space to vertex offsets and can be applied to an unseen actor at runtime.

2.2.3 Structured Light Approaches

Wang et al [55], in 2004, presented a method that uses high speed and high-resolution 3D dynamic data and fits a deformable mesh to the data in order to obtain good quality animations. Their approach is multi-resolution as they first track a low resolution mesh across frames by dividing the model into regions and controlling each region using a few parameters in order

to obtain coarse geometry. This coarse geometry is used to initialize the high-resolution mesh which is then fit to the frame data using non-rigid registration in order to obtain expression details. They then go on to present a method that learns a generative model from a set of facial expression performances from multiple people. This model facilitates the decomposition of facial expression into content of the expression vs individual style. Their method is based on the embedding of the facial expression manifolds non-linearly into a low dimensional space using Locally Linear Embedding, where all the facial expressions are normalized to achieve a 'unified embedding'. Non-linear mappings are then learned from this embedded space to the original expression space which allows for separation of personalized style. They go on to show examples of how this style can be transferred across individuals and also synthesize novel styles.

Zhang et al. [52] in 2008, presented a system comprising of 6 synchronized video streams — 4 monochrome and 2 color — combined with two video projectors that project grey-scale stripe patterns onto the face, in order to track a mesh consistently and with correspondences across the entire facial performance. A novel space-time algorithm is introduced which overcomes fitting deficiencies by making an assumption that disparity is nearly constant over a 3D space-time window and the depth is optimized over this entire window leading to improvements over standard stereo approaches. Their algorithm begins by fitting a template mesh to the first frame and then tracking the template through the sequence such that the shape matches the depth input and the vertex motions match the optical flow fields calculated between the frames of the color image streams. They then go on to show a real-time technique (faceIK) for editing the face to produce new expressions by exploiting correlations in a set of input meshes to propagate user edits to other parts of the face. Finally, they present a method to exploit the facial dynamics captured in their reconstructed sequences to produce tools for generating random facial sequences and for data-driven interpolation of user specified key-frames.

Li et al. [95] in 2009, presented a method that separates the large scale facial motion from small-scale facial dynamics. The make minimal assumptions about the dynamics of the motion and without requiring an underlying physical model or kinematic skeleton. A static acquisition method is used to reconstruct the initial template which is rigidly aligned manually with the first frame of the sequence. They propose a two-scale approach to reconstruct the sequence – a template registration stage captures the large scale motion by fitting a coarse template to every frame of the scan sequence while making use of detail coefficients estimated in the previous frame to enable locking and improving alignment accuracy, followed by a fine-scale detail synthesis step obtained by minimizing an energy resulting from point-to-point correspondences obtained between mesh vertices and scans along the vertex normals subject to a regularization constraint. In order to transfer details to occluded regions, they perform a separate pass that

aggregates detail coefficients using an exponentially weighted moving average which ensures that the influence of past detail decays quickly allowing for the handling of transient details like wrinkles. The detail aggregation is performed sequentially once forward and once backward allowing for back-propagating details seen in the future.

Weise et al. [53] in 2009, developed a system for live facial puppetry using a structured light scanner, thus enabling high-resolution and real-time facial expression tracking and also allowed for transfer of expressions onto another person's face. Their system consists of an offline phase where a generic template mesh is fitted to a sequence of expressions by the actor and a person specific linear model is learnt and an online phase where the reduced dimension linear space is used for online facial tracking and expression transfer. During the offline phase, a face model of the actor is built by having the actor turn his head in front of a scanner with a neutral expression after which a generic template mesh is fit to the reconstructed face with manually labeled landmarks for assistance. The actor performs 26 different facial expressions including long spoken sentences and the template is tracked through the entire sequence. This tracking stage involves optical flow for enhancing accuracy, a mouth segmentation stage for improved tracking around the mouth region, a rigid chin alignment step and separate eyelid tracking. Once the offline tracking is complete, Principle component analysis is used to obtain a reduced dimensionality and the face is manually segmented into subparts to remove global dependencies. Using this reduced space, the online tracking is achieved with around 15 fps. Finally, to enable facial puppetry, a linear subspace for the target face is calculated such that the face can be driven by the same coefficients as the actor's PCA model.

Ma et al. [3], in 2010, demonstrated the 'Digital Emily' project which aimed to cross the uncanny valley by creating a photo-realistic rendering of a 3D face captured using the USC ICT's Light Stage technology. The project succeeded in generating the first photo-realistic digital face to speak and emote convincingly at medium close-up, although the process involved a lot of manual interference. The actor's face was captured with 15 stereo photographs using off-the-shelf high quality still cameras under different illumination conditions in order to obtain the geometry and the reflectance information. Colored stripe patterns were projected onto the onto the face to facilitate robust pixel matching for the reconstruction. The skin texture detail is obtained by embossing the specular normal maps on top of the 3D mesh by minimizing the difference between the geometric normal maps and photometric specular normal maps. The actor's face is then scanned in 33 different facial expressions loosely based on the FACS coding system and these scanned expressions are used in a Blendshape framework. The obtained scanned and netural mesh are cleaned up manually to obtain a consistent re-topologized mesh. The face rig also contained displacement maps obtained from the original facial expression scans and these were blended in using the same weights assigned to the Blendshapes per frame.

This rig was fit to a monocular video of the actor. The 3D pose of the actor was set manually for multiple frames and optical flow was used to calculate the pose in between these frames. The lighting of the environment was obtained using a HDR light-probe. The face was re-lit and re-rendered into the video and a few edits were made to facilitate blending around the boundaries.

Ma et al. [71] in 2013, attempt to improve upon the 'Digital Emily' project with the 'Digital Ira' system. While 'Digital Emily' was rendered offline, it involved just the front of the face and was only medium close-up, in the 'Digital Ira' project, they aim to render in real-time, from any viewpoint, any lighting and even under very close range. This system was run in a production pipeline achieving 180 fps. The actor was scanned in 30 high-resolution expressions using the Light Stage X system in order to obtain 0.1mm resolution geometry and 4K diffuse and specular reflectance maps per expression. These expressions were merged into an artistically built back-of-the-head model. The actors lines and facial expressions were shot under seven views of 30 fps videos. They then use their Vuvuzela tool to interactively correspond all expression texture coordinates to the neutral expression, which was retopologized to a low-polygon mesh. Their offline animation solver creates a 'performance graph' from dense optical flow between the video frames and the expressions and then computes dense optical flow and 3D triangulation over the sequence and calculate the Blendshape weights per frame. Surface stress values are used to blend in the diffuse, specular normal and displacement maps from the high-resolution scans per vertex at run-time.

Fyffe et al. in 2011 [27] present a 'Comprehensive' facial capture pipeline and list out a few factors that they denote as essential for a comprehensive framework of facial capture — Dynamic capture, Full facial coverage, Detailed geometry, Detailed reflectance and automatic processing. They go on to present a system that allows reproduction of the performance which can be rendered from novel viewpoints, lighting conditions and photo-realistic rendering quality. Their system consists of five high-speed cameras placed in an 'M' formation in front of the actor allowing for full coverage and leeway for head movement. They capture the actor under active gradient illumination and from multiple cameras in order to estimate the reflectance function at each pixel. The estimated geometry is then merged with the reflectance data. This process is repeated for every frame of the sequence. Notably, they do not consider temporal correspondence in this system. Their present a novel heuristic for estimating detailed facial reflectance from gradient illumination photographs and a novel geometry optimization framework that maximizes a likelihood function that combines multi-view stereo and photometric stereo using a multi-resolution belief propagation approach.

Weise et al. in 2011 [96] presented a face tracking algorithm that combines 3D geometry and 2D texture registration with dynamic Blendshape priors obtained from existing facial animation

sequences. Their method works even in spite of noisy depth input data. A reduced facial expression model facilitates real-time processing. In an offline stage, the Morphable Model of [97] is used to register the mesh to the neutral face expression of the user in order to obtain the geometry. Facial expressions of the user are scanned and registered to the neutral mesh using a non-rigid registration approach with texture constraints around the mouth and eye regions to improve results. The user specific Blendshape set is generated automatically using the method of [17]. During runtime, unrealistic faces are prevented by regularizing the Blendshape weights with a dynamic expression prior computed from existing Blendshape animations from previous captures. A window of consecutive frames is considered in Blendshape vector space in order to exploit the temporal coherence of the animations. Their system makes use of a Mixture of Probabilistic Principle Component Analysis method in order to adequately capture the non-linear structure of the dynamic expression space, whilst also enabling real-time performance. The parameters of the MPPCA are estimated in the latent space of animations using a PCA and an Expectation Maximization framework.

Baltrusaitis et al. [98] in 2012, developed a 3D Constrained Local Model (CLM-Z) for tracking of facial features that integrates both depth and intensity into a unified framework. They also present a method to combine a rigid head pose tracker with their CLM-Z method and extend the generalised adaptive view-based appearance model (GAVAM) to use non-rigid tracking information leading to more accuracy. The use of depth data mitigates the effect of lighting variations and inconsistency which is a problem in purely video based methods such as the original CLM method. Their system adapts a two step CLM fitting strategy where an exhaustive local search is performed around the current feature point estimates leading to a response map for both intensity and depth features around each feature point and then iteratively updating the model parameters until convergence is reached.

Li et al. [30] in 2013, introduced a real-time capture framework that uses an adaptive PCA model that learns correctives on-the-fly specifically for the actor's facial expressions using incremental PCA based learning. This eliminates the need for a training phase or an offline phase where the system adapts to the user and the capture improves during the performance instead. Similar to [96], they obtain the neutral face shape by fitting a Morphable Model of faces to the neutral face accumulated scans and follow it up with a non-rigid deformation to obtain person specific details. They enhance this fit using 2D feature correspondences using the Live Driver software [99]. The initial Blendshapes, with crude approximations of the actor's expressions, are obtained using the deformation transfer algorithm like in previous works. During run-time, the Blendshape model is fit to the scans and followed by a Laplacian deformation algorithm, both making use of 3D and 2D point constraints. In order to train the correctives during run-time, they collect the new facial expression samples that fall out of the currently used adaptive

PCA space and these samples are used to add the corrective shapes incrementally to the adaptive PCA space using incremental PCA. These correctives adapt to the user's face and enable tracking with increased fidelity.

Bouaziz et al. [37] in 2013, attempt to consumerize facial performance capture by introducing a method that does not require any user-specific training or any manual assistance. Their system is quite similar to the one by [30] and jointly solves for both the 3D expression model of the user and for the dynamic parameters. Real-time performance is achieved by the use of a subspace parameterization. As more and more of the user's expressions are observed during the performance, the DEM progressively adapts to the user. A previously hand modeled template Blendshape model with the expression semantics that need to be transferred to the actor's model. A Morphable Model is used to obtain the neutral facial expression, after which a variant of deformation transfer is used for generating initial Blendshapes. Additional deformation fields are applied to the Blendshapes at run-time in order to obtain user specific details that the generic Blendshape does not capture. The optimization alternates between solving for rigid transformation parameters and Blendshape weights while keeping the DEM constant and refining the DEM by solving for the deformation coefficients keeping the Blendshape weights constant. Blendshapes that have been optimized sufficiently based on a threshold are removed from the optimization thus eventually converging to a final DEM.

Fyffe et al. [28] in 2014, use five high-speed cameras combined with gradient illumination patterns and obtain high-resolution face captures. Their system makes use of multiple high-quality static scans in order to account for the high-resolution details of the face. 30 scans of the actor's face are obtained including high-resolution geometry, specular and diffuse reflectance maps using the method of [54]. These scans cover facial expressions that roughly capture the whole expression space of the actor. Dense optical flow is used to obtain correspondences between the video frames and the high resolution scans and also between the neighboring video frames. Their system also computes a per-pixel confidence map for every optical flow based on Normalized Cross Correlation, ultimately generating a 'Performance Graph' whose edges reflect the dense 2D correspondences between all pairs of images and their weights reflect the level of confidence. These correspondences impose a weighted triangulation constraint between the static poses and the video frames. Their system is unique in that even partial correspondences yield drift minimization and this is reflected in the performance graph by having the confidence vary spatially over the image. Dynamic reflectance maps are achieved by blending in the static reflectance maps from the scans, after relighting the scans based on the lighting observed in the video.

Thies et al. [40], in 2015, presented a method for real-time facial reenactment using an RGB-D input device. Their method allows for an actor to control, in real-time, the facial expressions

of another person and overlay the re-rendered face into the original video to obtain realistic animations. They use a statistical model for the identity, expression and albedo of the face and optimize for the best model parameters in an analysis by synthesis approach. They model the scene lighting using spherical harmonic basis by assuming that the light source is distant and the scene is predominantly lambertian. They optimize for the parameters in real-time using a data parallel GPU solver. Their system re-renders the mesh after relighting it and seamlessly overlays it on top of the original video leading to expression transfer onto the target actor's face. Their method works in real-time and obtains very realistic results. The emphasis of their method is less on 3D geometrical accuracy and more on obtaining visually plausible results that can be re-rendered on top of the original face in the video video such that the face can be reanimated and controlled in an indistinguishable and realistic fashion.

2.3 Conclusions and Motivation

As discussed, the body of work in the area of Performance Driven Facial Animation is vast and has spanned multiple decades. While there has been tremendous progress that has been made towards the goal of achieving realistic facial tracking, modeling, retargeting and display, there are still gaps that need to be filled, especially when it comes to the goal of a system which combines realistic animations, operates in real-time, is automated, cheap and adapts to individual faces without the need for cumbersome training phases. e.g. the Modeling phase of Performance Driven Animation is one of the areas that involves the most manual interference from 3D modelers in order to obtain realistic face meshes with useful topological properties such as properly spaced and evenly distributed vertices and contours. This becomes even more challenging when trying to personalize Blendshapes to a specific individual. The use of automated methods to do this alleviates this to an extent but as is expected, a lot of individual nuance is lost as these methods usually transfer details from an underlying template model which is unaware of these nuances. This means that a 3D artist has to add these details in a post-process step. In our work, we address this area and provide an automated method of generating a full animatable Blendshape Rig from a single scan or a few photographs of an individual. We also present our approach to adding in personalized nuances to Blendshape expressions. There has been ample use of sparse landmark points in order to aid the Capture stage of facial animation including both markers and automatically detected features from video. On the other end of the spectrum, the use of dense texture has also been successfully shown to produce excellent results while making use of all the texture information available. Both these approaches have their advantages when it comes to ease vs accuracy. An optimal combination of these two approaches in order to leverage the best of both worlds is an area that is still relatively less explored. Making use of localized texture regions on the face and combining it

with sparse landmarks is something we explore in this work. One of the goals of Performance Driven Facial animation is for the pipeline to be consumer friendly. This implies complete automation and the absence of cumbersome training phases, especially one's that require hard to obtain training data or for the actor to provide multiple training performances beforehand. While there has been a clear trend of improvement in this regard, there is still room to make the capture process more accessible to the average consumer and this is one of the motivations in this work. In the next sections, we present our contributions to the Modeling and Capture phases of the Performance Driven Animation pipeline.

3 Modeling - Representation of Digital Faces

3.1 Overview

One of the important considerations when doing Performance Driven Facial Animation is how we represent the digital face and how we animate it. While a clear delineation can be made between the capture method and the representation method, the choice of representation does have an effect on the final animation in terms of how much of the captured information can be faithfully recreated. Over the course of a few decades many approaches to how digital faces are represented and animated have been proposed and each of these have their pros and cons.

In recent years, there has been an emergence of work that captures 3D faces at a high fidelity based on mesh propagation. In this case, a single mesh of the target's face is deformed over time in order to generate an entire capture sequence [8, 25, 26, 27, 5, 28, 93]. The strengths of this approach is that the underlying representation of the face is not limited within a space and given accurate capture parameters, it can span the full space of motion of the actor in question. While this method is relatively straight-forward and can capture very high resolution data, this is difficult to animate or modify later on by an artist as the capture is usually in the form of vertex displacements on the mesh and does not provide a meaningful *parameterization* of the face. This also restricts immediate use later on, e.g. for facial re-targeting to a second model with a parallel parameterization. From the perspective of a 3D animator, the vertex motion over time is not intuitive enough to manipulate and adjust and usually they prefer control mechanisms that have a semantic meaning with regards to facial expressions.

An alternative method for facial representation involves a parametric model of a face. This first requires an appropriate parameterization, and various approaches have been proposed. Statistical models, based on e.g. Principle Component Analysis (PCA), are a convenient means of providing an orthogonal basis of facial expressions [97, 29, 31]. The drawback with PCA is that individual modes generally do not reflect meaningful or useful facial shapes and don't lend themselves to semantically meaningful interpretation. This makes them inconvenient for later modification by an artist, or for re-targeting onto other PCA models where the facial expression basis would generally differ. PCA is very convenient when it comes to reducing the dimensionality of the expressions space and the individual basis can be obtained automatically from the actor's performance itself, but there is no guarantee of what each basis of the model represents, which can cause unintuitive mixing of multiple facial expressions within a single basis.

On the other hand, Blendshape based linear models are a more common type of facial model

used in animation [37, 36, 17, 30, 33, 38]. In this approach, each individual basis of the parametric model represents a facial expression represented as a bunch of vertices and triangles which are in correspondence across the expressions. These expressions/bases are then combined linearly in order to generate new facial expressions. This representation is particularly very intuitive as each basis corresponds to a semantically and physically meaningful facial expression. This makes it very easy for 3D artists to create and modify these bases. Although the model itself is very simplistic and the space of facial expressions is limited by the choice of basis, in practice it leads to very visually convincing animation and is simple enough to be applied in real-time. Excellent in-depth recent surveys in the area of Blendshape models, facial rigging and facial model representation may be found in [100] and [101].

Muscle based anatomical models combined with polygonal skin surfaces have been used in the past [6, 43, 44, 45, 46, 47] but seem to have lost favor recently. These muscle based models require considerable effort to create – an anatomical model of muscles represented by splines or polygons with a polygon mesh as a skin surface on top – and usually require non-linear activations in order to produce good results [6]. While this method is physically more accurate, the effort involved in creating the model combined with the complexity of the solver has made Blendshape models the preferred choice of representation within the VFX industry.

In our work, we make use of Blendshape representations for faces. Traditionally, the creation of Blendshape models specific to an actor's face used to be a very skilled and time intensive task reserved for 3D Modelers. Recently though, many methods either based on statistical approaches [97, 32] combined with deformation [16] have been proposed that makes the generation of actor specific Blendshape models a very automated procedure which simplifies the facial animation pipeline significantly. In the next few sections, we show how we use an existing template Blendshape model to automatically generate personalized Blendshapes for a new user saving considerable time which would otherwise be lost in creating these manually. We then discuss methods to automatically generate Blendshapes that capture the actor specific nuances which the template Blendshape model might not capture.

3.2 Generating the Neutral Face Mesh

The first step in generating Blendshapes is to obtain a 3D mesh of the actor's face with appropriate geometry i.e. the required number of vertices and triangles. While scanning the actor's face will provide a 3D mesh, usually the topology of this mesh is undesirable as it is noisy, has random number of vertices and triangles and may have holes in the mesh as shown in Figure 10. This needs to be processed in order to generate a 3D mesh that has the desired topology but maintains the structure and shape of the actor's face. In the next few sections, we discuss how

we obtain this desired mesh using a scanning pipeline followed by non-rigid registration methods and then go on to present an alternative approach that builds upon the scanning pipeline and further automates this process by making use of a statistical model of faces in order to automatically generate this mesh from 2D images of the user.

3.2.1 Blendshape Equation

The basis of our Blendshape model generation pipeline is an existing template model created by a professional artist. We describe this model using the standard delta form [100], as follows:

$$\mathbf{A} = \mathbf{A}_0 + \sum_{i=1}^{N} \alpha_i (\mathbf{A}_i - \mathbf{A}_0) \tag{1}$$

where $\mathbf{A} = [x_1 y_1 z_1 \ldots x_n y_n z_n]^T$ is a vector of n vertices representing the target face, $\mathbf{A}_0$ is the neutral facial mesh, $\mathbf{A}_i$ is one of N Blendshapes and α_i is the Blendshape weight.

In our existing generic model, there are $N = 140$ Blendshapes. The Blendshapes in this generic model have the desirable property that the mesh topology contains edge loops around natural facial contours and has a smooth surface as shown in Figure 11.

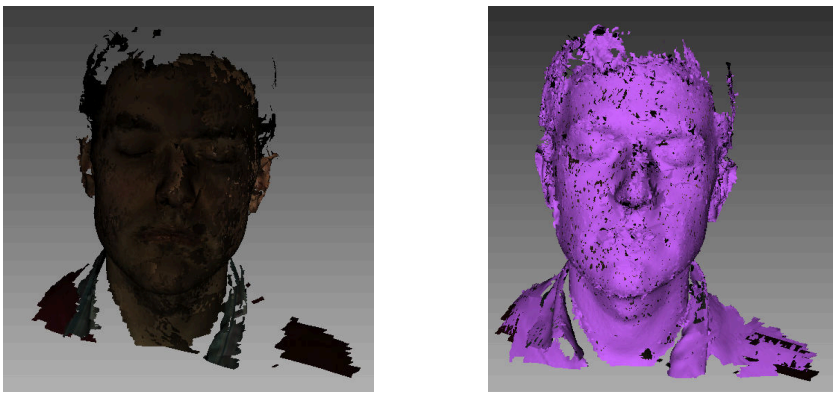


Figure 10: (a) Raw scan with texture, obtained from the 3D scanner [10]. (b) Raw mesh without texture.

Creating a personalized model first requires the creation of a new Blendshape neutral expression $\mathbf{B}_0$ with the same topology. We use the method of [11] for non-rigidly registering the template neutral mesh and target 3D scan. While this approach can operate without correspondences, a higher quality registration can be obtained by supplying correspondences between the source and target meshes. This effect can clearly be seen in Figure 12.

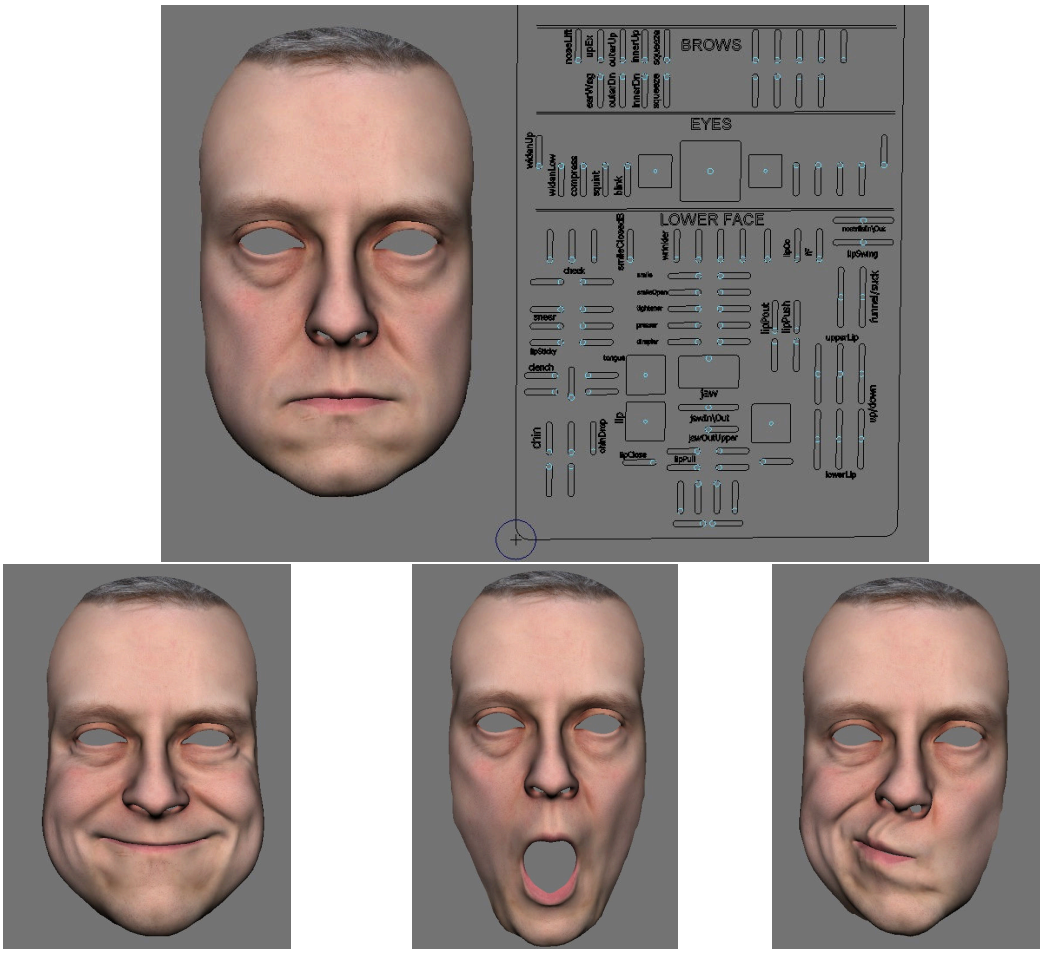


Figure 11: Our system leverages an existing generic 140 expression Blendshape model encapsulated in a *Maya* interface.

3.2.2 Automatic Correspondence Detection

In order to automatically detect corresponding landmarks in UV-space for a given face mesh, we propose a HOG [63] based feature detector. Alternatively the landmarks can also be detected using recent approacehs like [102, 103]. In addition to these, we also generate correspondences around the inner mouth and eye regions (which are usually the challenging areas to register correctly) by automatically tracking a curve along the inner lips and eyelids in UV-space. See Figure 13 and 14.

We first generate an annotated dataset where 12 regions in the UV-space of the face were manually marked with rectangles centered on the region of interest i.e. feature points. The chosen feature points were the ”Nose”, ”Between the Eyes”, ”Right Eye Right Corner”, ”Right Eye Left Corner”, ”Left Eye Right Corner”, ”Left Eye Left Corner”, ”Left Cheek”, ”Right

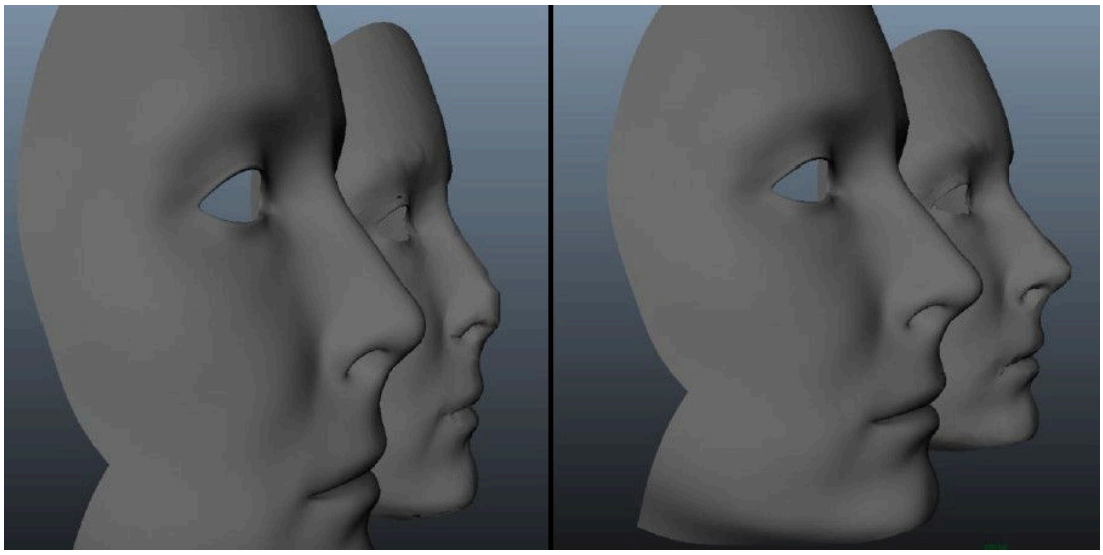


Figure 12: Effect of using landmark correspondences in the non-rigid ICP algorithm. (Left) the nose tip on the deformed mesh is flatter compared to the ground-truth. (Right) The use of landmarks on the nose improve deformation results.

Cheek", "Left Mouth Corner", "Right Mouth Corner", "Top of Mouth", "Bottom of Mouth" (See Figure 13). A subset of 30 images were used for training.

The training procedure was as follows:

Each image was first resized to be 1024x1024 and unsharpened using a Gaussian filter. For each of the images and each landmark in the image a HOG feature of size 32 was extracted. The same filter was then applied to every point in the image and if the resulting HOG feature was within a similarity threshold of the landmark in question, it was retained. On top of this if a feature lay within a 20 pixel radius of the centroid of the rectangle identifying the landmark, the matching score was recorded as good, otherwise it was recorded as bad.

We then fit a Gaussian distribution for bad scores and the good scores as $\mathcal{N}(\mu_{good\_scores}, \sigma_{good\_scores})$ and $\mathcal{N}(\mu_{bad\_scores}, \sigma_{bad\_scores})$. Furthermore, we modeled the prior of a feature being at a particular location as a multi-variate normal centered at the mean location across all images.

So e.g. if a landmark is located at (x,y). Then any pixel within (x-20,y-20), (x+20, y+20) is considered to be the "right location". This allows for a little noise. Presumably things around x,y will be similar. Now, when we apply the filter at each point within this rectangle it will produce a scalar. The set "Good Scores" contains all such scalars. However, we also convolve the filter with pixels outside the "right location". Every scalar from this is put into the "Bad

Scores" set. We do this for every image in the training set. The reason we do this is because we don't know a priori how well each filter we extracted from the training set will match a new face. So we want to get an idea of what it means for a filter to be a good match. If the distributions of "Bad Scores" and "Good Scores" overlaps too much then this filter will not be a good discriminator. Given a new image we apply the filter to each pixel and then compare that value with the "Good Scores" set and the "Bad Scores" set. If that value is more likely to have come from the "Good Scores" set then we keep it. If it is more likely to have come from the "Bad Scores" set then we discard it.

Given a new UV-space image, we convolve each filter with it and the top scoring 200 features are retained. The features that have scores that were more likely to have come from the bad-scores model were removed. The remaining detected locations were clustered using the mean-shift clustering algorithm [104] with bandwidth 10. Each cluster was then scored by multiplying the size of the cluster with the prior on location for that feature and the top scoring cluster center is marked as the desired landmark.

For the eyes and the mouth, the corner landmarks detected from the previous step, i.e. the left-most and right-most corners, are used as a starting point in order to trace a curve along the contour of the eyes/lips. A bounding box (with +-50 pixel border around the center) was defined around the corner landmarks. An algorithm inspired by [105] was then used to identify the line separating the upper and lower portions of the eyes and mouth. This line was then traced subject to constraints that

1. the adjacent points have similar colour/intensity
2. the overall intensity of the curve is lowered
3. and curve smoothness is preserved.

We optimize this line using the Iterative Conditional Modes algorithm. This algorithm starts from one corner and traces the curve along the contour to the opposite corner by making locally optimal decisions at each pixel. An initial estimate is given by a line joining the left and right corners. Starting from the left-most corner it picks the next pixel in the image from the adjacent column to the right while optimizing for all 3 criteria and then moves on to the next column. Any points that have intensities greater than 1 standard deviation from the intensities of the current pixels in the contour are discarded. As this is a local decision, we do not obtain the solution in the first pass. The algorithm is iterated multiple times until it converges or a threshold is reached. Once the contour is traced along the eyes and mouth, we select evenly spaced pixels on the contour and obtain the corresponding 3D point on the face mesh by mapping the pixel to the closest point in UV-space, that corresponds to a 3D vertex on the face. This is done

for both the template mesh and for the target mesh in normalized UV coordinates. An example of the landmark points in UV-space and in 3D can be seen in Figure 14. These automatically detected landmarks are then used by us for correspondences in the Non-rigid ICP algorithm.

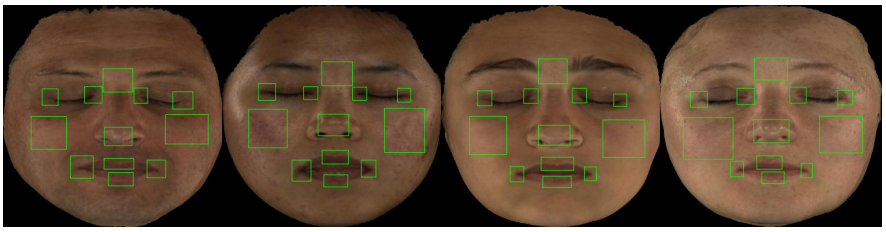


Figure 13: We train a HOG based feature classifier in UV space that is trained to detect landmark points on the actor's face. We train our classifier on 30 faces.

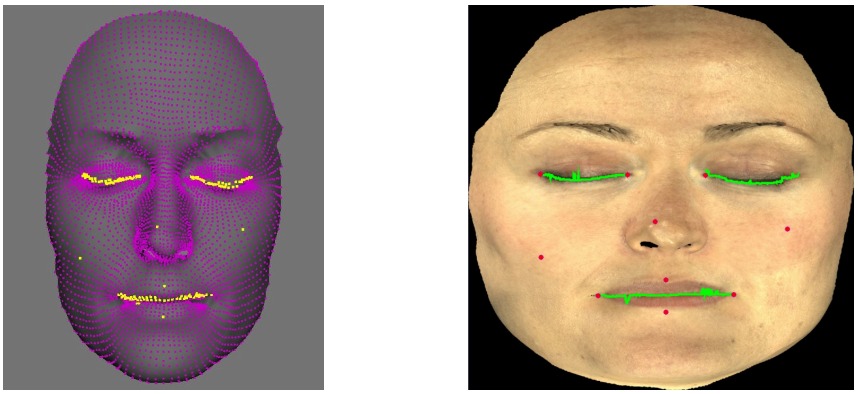


Figure 14: We generate a large number of correspondences around the inner mouth and eye regions by automatically tracing a curve along the inner lips and eyelids in UV-space.

3.2.3 Non-rigid Registration

The target mesh $\mathbf{B}_0^{'}$ is obtained by scanning a participant in a neutral expression with their eyes closed using an Artec Eva scanner [10]. In its current form, $\mathbf{B}_0^{'}$ contains a different topology from $\mathbf{A}_0$. Using the landmark correspondences to assist the non-rigid registration, we generate a personalized neutral mesh $\mathbf{B}_0$ with same topology as $\mathbf{A}_0$ as shown in Figure 17.

The cost function for the non-rigid ICP algorithm of [11] can be described as follows:

$$E(X) := E_d(X) + \alpha E_s(X) + \beta E_l(X) \tag{2}$$

where the terms $E_d(X), E_s(X)$ and $E_l(X)$ are the data-term, the stiffness term and the landmark term respectively, and α, β are the respective weighting terms.

The term E_d is given as follows:

$$E_d(X) := \|W(DX - U)\|_F^2 \tag{3}$$

where $W := diag(w_1, ..., w_n)$ is a weighting matrix used to weight the importance of the vertices to the deformation. The sparse matrix D is defined as

$$D := \begin{bmatrix} v_1^T & & \\ & ... & \\ & & v_n^T \end{bmatrix} \tag{4}$$

where $v_1, ..., v_n$ are the known vertex locations on the mesh to be deformed and $U := [u_1, ..., u_n]^T$ are the corresponding points on the target mesh obtained using a KNN search for closest match. X contains the unknown transformation matrices X_i applied to each vertex v_i.

The stiffness term E_s penalizes the differences between the transformation matrices assigned to neighboring vertices, given by

$$E_s(X) = \|(M \otimes G)X\|_F^2 \tag{5}$$

where M is known as the node-arc-incidence matrix and contains one node for each edge of the the mesh and one column per vertex. If edge r connects the vertices (i, j), the non-zero entries of M in row r are $M_{ri} = -1$ and $M_{rj} = 1$. The term G is given as: $G := diag(1, 1, 1, 1)$. The symbol $\otimes$ represents the Kronecker product.

The landmark term E_l is similar to the distance term and is given by

$$E_l = \|D_L X - U_L\|_F^2 \tag{6}$$

where D_L contains the rows out of D that correspond to the landmark vertices and similarly for U_L.

Finally, the complete cost function is given by

$$E(X) = \left\| \begin{bmatrix} \alpha M \otimes G \\ WD \\ \beta D_L \end{bmatrix} X - \begin{bmatrix} 0 \\ WU \\ U_L \end{bmatrix} \right\|_F^2 = \|AX - B\|_F^2 \tag{7}$$

This can be minimized by setting it's derivative to zero and solving the resulting system of linear equations. $E(X)$ takes on it's minimum at $X = (A^T A)^{-1} A^T B$. Here α represents the stiffness weight that controls how much the stiffness term affects the output. Similarly β

controls the influence of the landmark points. Please refer to [11] for further details.

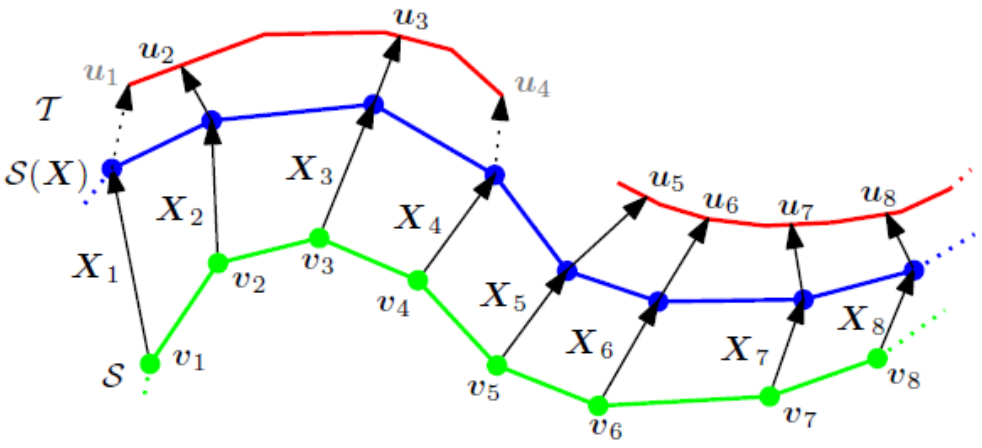


Figure 15: [11] Diagram showing the terms involved in equation 3. The template surface S(green) is deformed by locally affine transformations (Xi) onto the target surface T(red). The algorithm determines closest points (u_i) for each displaced source vertex ($X_i v_i$) and finds the optimal deformation for the stiffness used in this iteration. This is repeated until a stable state is found. The process then continues with a lower stiffness. Due to the stiffness constraint the vertices do not move directly towards the target surface, but may move parallel along it. The correspondences u_1 and u_4 are dropped as they lie on the border of the target.

Algorithm 1 Non-rigid ICP

1: Initialize X_0 to have the identity matrix for each vertex.
2: Initialize the node-arc incidence matrix M based on the edge connectivity
3: **for** each stiffness $\alpha^i \in \{a^1, ..., a^n\}, a^i > a^{i+1}$ **do**
4: **while** $\|X^j - X^{j-1}\| < \varepsilon$ **do**
5: Use the KNN algorithm to find preliminary correspondences for VX^{j-1}
6: Determine X^j as the optimal deformation for the correspondences and α^i

It consists of 2 loops. The outer loop finds a series of deformations that bring the template closer to the target while reducing the stiffness gradually allowing for more localized transformations. The inner loop is where a deformation is found for a fixed stiffness. The algorithm assumes a reasonable initial rigid alignment between the template and target mesh before being applied. This can be done using a simple Rigid ICP algorithm as shown in Figure 16.

Using this algorithm, we obtain our neutral face mesh B_0 which we use in the next step of the pipeline. Results from this algorithm can be seen in Figure 17 and 18.

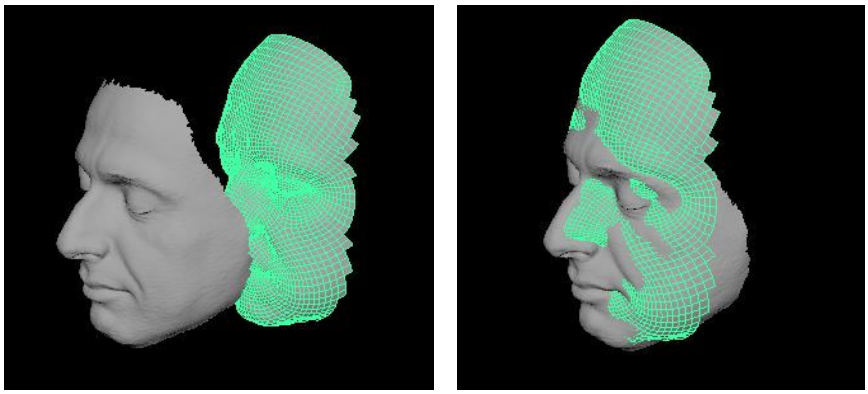


Figure 16: Initial rigid alignment performed between the template mesh and target scan before doing a non-rigid alignment.

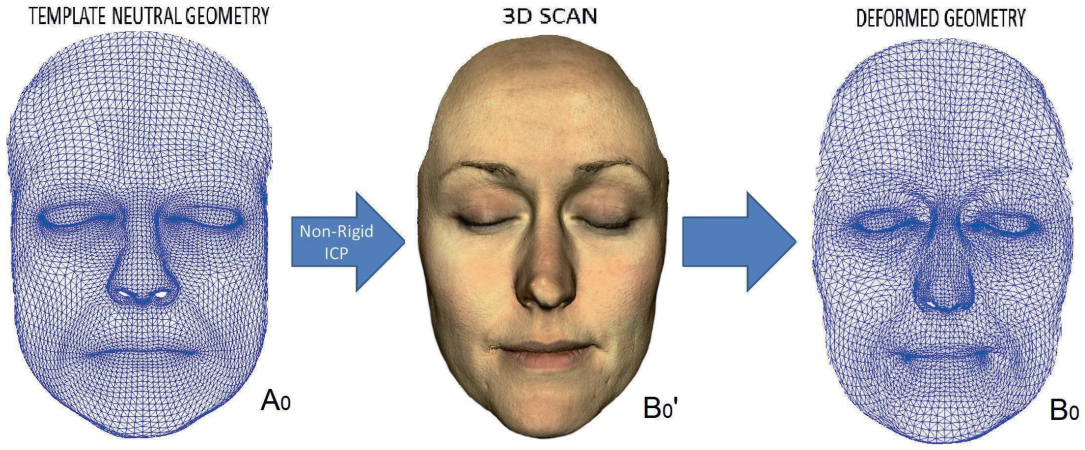


Figure 17: The template mesh neutral geometry is deformed non-rigidly towards the 3D scan, resulting in the deformed geometry.

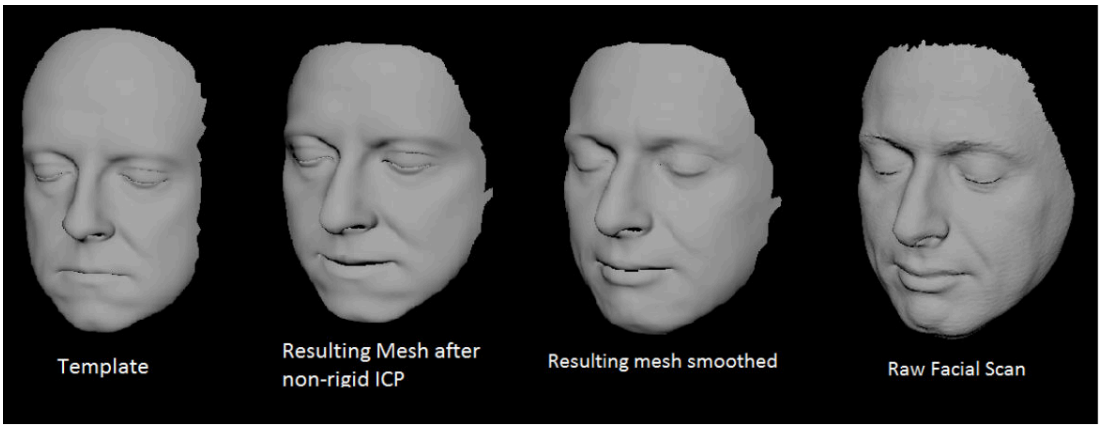


Figure 18: Example showing the application of algorithm (1) to our template and scanned mesh.

3.2.4 Statistical Model For Neutral Face Meshes

While the use of scanners [10] followed by a non-rigid ICP as discussed in the previous section is an acceptable way of obtaining a neutral face mesh in a desired topology, the prohibitive costs of scanners might be an issue. Of course methods like [106] that make use of multiview stereo approaches in order to obtain high quality geometry are another way to obtain a neutral face mesh. This again needs to be followed up by the non-rigid warping to obtain desired topology. Many a breakthrough in this sector comes as a result of advances in the scanning hardware, largely through improved speed and accuracy. The variety of scanning methods are plentiful and each have their strengths. [12] propose the taxonomy for the various optical acquisition techniques as shown in Figure 19.

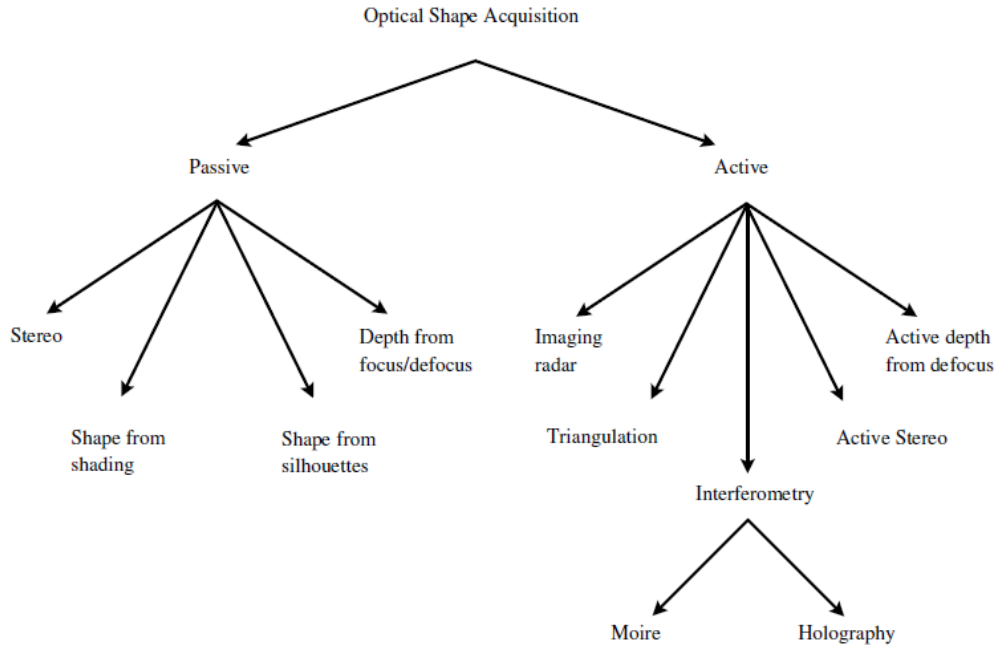


Figure 19: Taxonomy of various optical acquisition techniques [12].

[97] proposed a 3D Morphable Model of faces which is essentially a statistical model of faces created from multiple laser scans of around 200 individuals which are then put in correspondence with each other. They also propose an algorithm by which, given an image, or a few images of an individual, the model can be fit to the images in an analysis-by-synthesis approach that minimizes the difference between the synthesized rendering and the original image. This is very cost effective as it avoids the use of scanners and also avoids a non-rigid warping step. This can save considerable time and effort when the need for obtaining the neutral face mesh for many individuals arises. This is also achievable in real-time as demonstrated by [40, 35]

making it an extremely useful step to have in the facial animation pipeline. In this section we discuss our efforts to create a statistical model of human faces in order to further improve our Blendshape creation pipeline.

The 3D Morphable Model can be described as follows: The geometry of each face in the model can be represented by a shape vector $S = (X_1, Y_1, Z_1, ...X_n, Y_n, Z_n)^T \in R^{3n}$ that contains the X, Y, Z coordinates of its n vertices. Assuming that the number of texture values are equal to the number of vertices, we can represent the texture of the face as a vector $T = (R_1, B_1, G_1, ..., R_n, B_n, G_n)^T \in R^{3n}$, that contains the R, G, B color values of the n corresponding vertices. A morphable face model can then be constructed using a data set of m exemplar faces, each represented by its shape-vector S_i and texture vector T_i. New shapes S_{model} and new textures T_{model} can be expressed in barycentric coordinates as a linear combination of the shapes and textures of the m exemplar faces.

It is important to be able to quantify the results in terms of their plausibility of being faces. The probability distribution for the coefficients of the model are obtained from the example set of faces. This distribution enables us to control the likelihood of the coefficients and consequently affects the likelihood of the appearance of the generated faces. Thus the generated face does not deviate far from the space of physically plausible faces.

Principal Component Analysis is then used in order to generate a set of orthogonal bases which can be combined in order to generate new faces. The model can be specified as follows:

$$S_{model} = \overline{S} + \sum_{i=1}^{m-1} \alpha_i s_i, \quad T_{model} = \overline{T} + \sum_{i=1}^{m-1} \beta_i t_i \tag{8}$$

where, $\overline{S}$ and $\overline{T}$ are the average of the shape and geometry vectors and s_i and t_i are the eigenvectors of the covariance matrices C_s and C_T computed over the mean centered shape and texture data. The model is parameterized by the coefficients $\overrightarrow{\alpha} = (\alpha_1, \alpha_2 ..... \alpha_m)^T$ and $\overrightarrow{\beta} = (\beta_1, \beta_2 ..... \beta_m)^T$ where $\overrightarrow{\alpha}, \overrightarrow{\beta} \in R^{m-1}$. The probability distribution over $\overrightarrow{\alpha}$ is given by:

$$p(\overrightarrow{\alpha}) = \exp[-\frac{1}{2} \sum_{i=1}^{m-1} ((\alpha_i)/(\sigma_i)^2] \tag{9}$$

where σ_i is given by the eigenvalues of the shape covariance matrix C_S . The probability $p(\overrightarrow{\beta})$ is computed similarly.

In order to obtain our database of faces, we scanned 30 different people using our Arctec [10] medium resolution scanner. We then used the algorithm of [11] in order to deform a template

neutral face mesh with desired topology to our scanned faces, as discussed in Section 3.2.3. Note that this is different from the way the original authors get the faces in correspondence using optical flow [97]. While we scanned our own set of faces, one can also make use of the original set of 200 faces used in by [97]. Also recently, [32] created a Morphable Model from 10000 face scans and tailored for specific age, gender or ethnicity groups. Once we do this, we have all our faces in correspondence — i.e. with the same number of vertices and triangles. We then build our morphable model as described above in equation 8. This gives us a statistical model of faces which can be used to generate a new 3D face (with desired topology), given a few images of any person. This procedure is described in the next section.

The results of our Morphable Model generated from our scanned faces can be seen in Figure 20 and 21. Figure 22 shows an example of a face generated when we vary the value of the coefficients beyond ± 3 standard deviations from the mean. Figure 23 shows an example of linear morphing between 2 faces in the model.

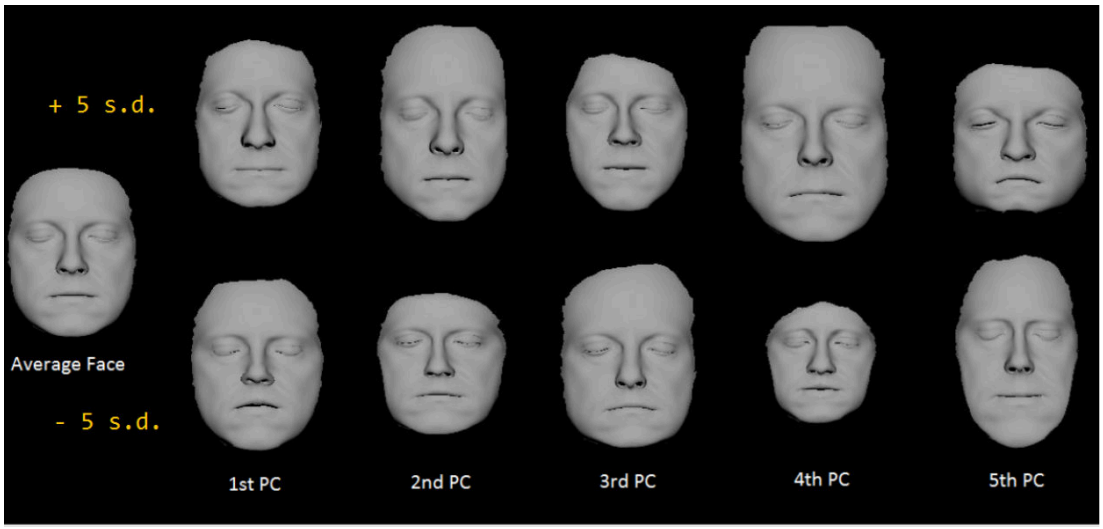


Figure 20: Results obtained by varying the first 5 principal components of our 3D Morphable Model.

Figure 21: Random faces generated from our 3D Morphable Model.

Figure 22: Example of an out-of-space face generated using our 3D Morphable Model when we vary the value of the coefficients significantly beyond ± 5 standard deviations from the mean.

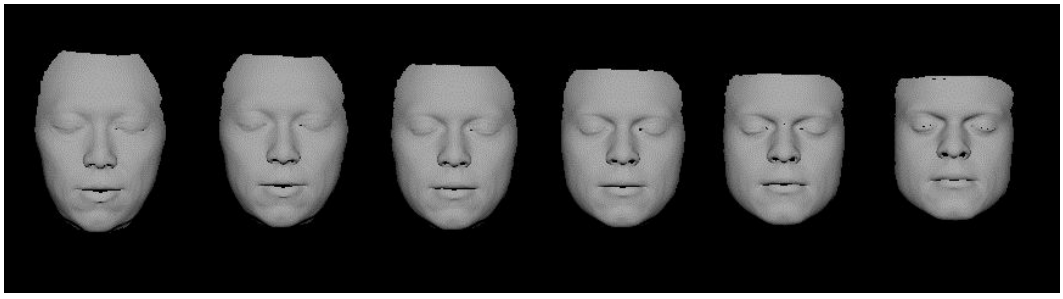
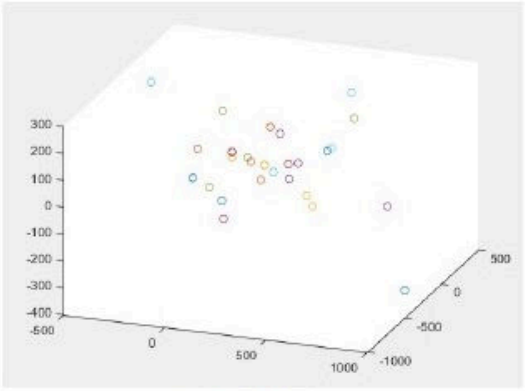
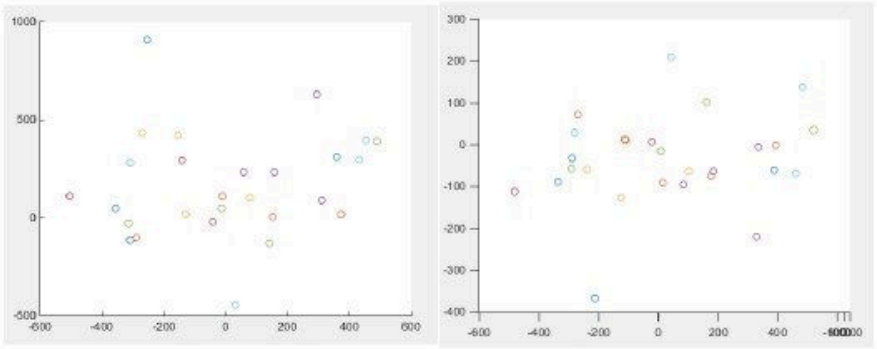


Figure 23: Image showing results of morphing between 2 faces using the 3D Morphable Model. This shows the variation in space along these axes that our model spans.



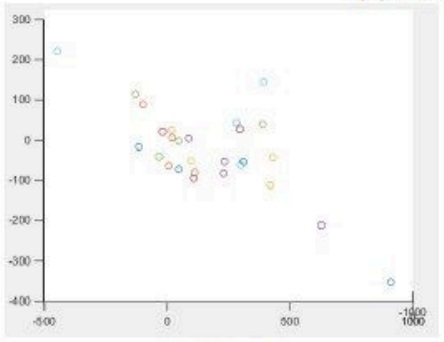
(a) Identity

Figure 5.6: First 3 PCs for facespace



(a) PC1 v PC2

(b) PC1 v PC3



(c) PC2 v PC3

Figure 24: Plots of the first 3 Principal Components of our face data used to build the 3D Morphable Model.

3.2.5 Fitting a Morphable Model to an Image

Given the 3D Morphable Model, our objective is to generate a 3D shape for the actor's face, with an image or a few images of the actor. This enables us to obtain the 3D shape of the actor in a relatively inexpensive way compared to manually scanning the actor and then performing the non-rigid ICP using a template face geometry. In order to do this, our optimization framework involves simultaneously optimizing for the shape parameters in the 3D morphable model and the expression parameters for the expression of the actor in the image, in an iterative way. Our Blendshape model is a pre-existing model created by an artist as shown in Figure 11 and as represented in equation 1. The coefficients of this Blendshape model are constrained between 0 and 1. Given an image (or multiple images) of the user, the optimization iterates between solving for the coefficients of the morphable model and the coefficients of the Blendshape model. This is because the image of the user can have him or her with a non-neutral facial expression. We make the assumption that the actor has the same expression if multiple images are provided. Using multiple images, especially profile pictures of the actor, help with resolving depth ambiguities, such as how pointy the nose is, that might not be apparent from a single picture.

The process of using a statistical model in order to generate 3D faces of users can be split into two main sub-problems — Creating a 3D morphable model and optimising the parameters of the 3D morphable model to match a 2D image with the goal of creating a new 3D face with the desired topology.

Within these two sub-problems, there are two themes. The shape of a face, including the facial expression, and the texture applied to that face. As a result, the process can be split into two streams which can be combined at the end.

For the 3D shape of the face, the algorithm can be listed as below:

Algorithm 2 Generating the geometry for the face

1: Generate the 3D morphable model for the shape
2: Automatically detect landmarks on an image of the user
3: Estimate camera pose for the given image
4: Specify a Blendshape model for facial expressions
5: Specify the cost function for the shape
6: Optimize the parameters of the model to generate the shape
7: Optimize the parameters for the facial expression
8: Iterate 6-7 until convergence

In order to extract the texture for the model from the images, the process is:

Algorithm 3 Generating the texture for the face

1: Generate the texture morphable model
2: Specify the cost function for an image
3: Optimize the parameters of the texture morphable model

We now describe the process in detail.

LANDMARK DETECTION

Given the images of the actor, it's necessary that we detect sparse landmarks on the actor's face before we can optimize for the parameters of our model. Once we obtain these, the corresponding points in 3D are marked by the user in a one time step and these correspond to vertices in the 3D Morphable Model, with semantically corresponding locations on the face. In our experiments, we use the algorithm proposed by [14] (Chehra) in order to automatically detect landmark points in the image. This algorithm detects the face within the image and uses an iterative method to find 49 landmark points on the eyes, nose and mouth. This is done by initially using the method of Viola-Jones [13] for face detection, to create an object bounding box surrounding the face.

The method of Viola-Jones [13] is one of the most efficient methods of face detection. It combines a selection of weak binary classifiers, such that each has little over 50% classification rate, to produce a far more efficient classifier. They use a selection of Haar-like features which, when calculated with the integral image, can be computed in constant time. Haar-like features are the results of the difference between the sum of two sets of pixels, these show light and dark areas of an image, the shape of which come from features such as eyes and eyebrows. Figure 25 shows some examples of Haar-like features, in which the sum of the pixels in the white rectangle are to be subtracted from the sum of the pixels in the gray rectangle.

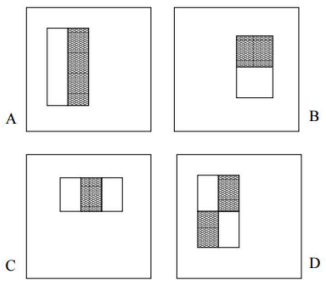


Figure 25: Haar features used for a boosted face detection algorithm [13].

The Haar-like features each act as a weak classifier, describing features that are present within images of faces. The object bounding box surrounding the face provides an initial landmark

location, s^a. A function, g, is trained such that it can map the initial shape from the object bounding box of image I_i, to the ground truth image s_i^*, given a set of features of the image, f.

$$g(s^a, I_i, f) = s_i^* \tag{10}$$

The features of the image are found by concatenating the results of SIFT, trained using a set of images with the feature points pre-defined. The shape model used by [14] is a 3D parametric shape model given by

$$s(p) = sR(\overline{S} + \phi_s g) + t \tag{11}$$

The above model is trained using a parallel cascade regression function for the set of parameters $p = s, R, t, g$, where s is taken from the set of training faces. Please refer to [14] for further details.

Figure 26 shows an example of Chehra [14] in use, the red points show the initial estimate of landmark points. The green points show the final estimate of the landmark points. Using this method ensures that the points detected are consistent and can be identified on the morphable model. However, this does not allow the shape of the jaw to be modeled. To do this, a selection of 19 user defined points have been chosen, with a sub section of these points to be used depending on the images used. This allows for off center images to be used to help to obtain the depth of the model.

Figure 26: Landmark features detected on a face using the algorithm of [14].

CAMERA PROJECTION ESTIMATION

In order to be able to estimate the error in our objective function, we need to project the 3D points onto the image. This involves calculating the camera projection matrix. This is known as the Perspective-n-Point problem. In our experiments, we use the method of [15], which provides a solution to the Perspective-3-Point (P3P) problem as shown in Figure 27.

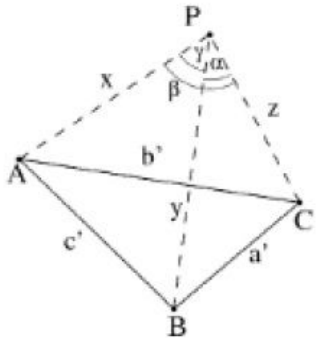


Figure 27: Perspective 3 Point Problem [15].

An implementation for the Perspective-n-Point problem using the method of [15] for MATLAB, *estimateWorldCameraPose()*, has been provided by MathWorks Documentation (2017b). This provides an estimation of the location and orientation of the face with respect to the camera, with a set confidence measures and maximum error in the projection.

Given the translation t and the rotation R, we can then specify the projection matrix P of the camera that projects the 3D points into the image as below:

$$P = K[R|t] \tag{12}$$

where K is the intrinsic matrix of the camera. In our experiments, we assume a focal length of 6000 and assume no radial distortion or skew.

OPTIMIZING FOR SHAPE PARAMETERS

Now given the camera's projection matrix, the 2D landmarks on the images and the corresponding 3D points, our objective function takes the form of a sum-of-squares error, where we minimize the squared distance of the projected 3D points and the corresponding detected 2D landmarks, over all the landmarks and over all the images.

Assuming that we have 3 images of a user's face in a left profile, central/front-on view and right profile, our objective function looks as below

$$E_{Fit} = E_L + E_C + E_R \tag{13}$$

where E_L, E_C and E_R correspond to the energy associated with the left-profile, center and right-profile images of the user.

The energy associated with each of these images is given as

$$E_L = \sum_{l=1}^{n} \|P(\overline{S} + \sum_{i=1}^{m-1} \alpha_i s_i)^{(v_l)} - q^{(l)}\|^2 \qquad -3\sigma_i \leq \alpha_i \leq 3\sigma_i \tag{14}$$

where:

- n is the number of landmarks
- P is the camera projection matrix
- m is the number of eigenvectors
- $\overline{S}$ is the mean shape vector
- α_i is the weight associated with eigenvector i with the constraint as in equation (9)
- s_i corresponds to the i-th eigenvector
- $q^{(l)}$ represents the l-th 2D landmark point in the image
- v_l represents the vertex corresponding to landmark l
- σ_i is the standard deviation along the PCA dimension i. This constraint is natural as the weights should ideally not deviate from the training data and as the PCA implicitly implies a Gaussian fit to the data, 3 standard-deviations cover 99.7% of the training data.

This is a nonlinear constrained optimization problem, which can be solved using the fmincon function in MATLAB. The process of estimating the camera position and optimising the parameters for the 3D morphable model is iterated until convergence. This is to reduce the error in camera rotation matrix and translation vector as this is estimated using the current best 3D model and so is not exact.

OPTIMIZING FOR FACIAL EXPRESSION PARAMETERS

Once we have the shape parameters of the Morphable Model, we then need to solve for the facial expression of the user in the image. Our expression model uses a Blendshape basis and the optimization is done in order to minimize the squared error between projected points and detected points as before. It can be stated as follows:

$$E_L = \sum_{l=1}^{n} \|P(B_0 + \sum_{i=1}^{N} \alpha_i B_i)^{(v_l)} - q^{(l)}\|^2 \tag{15}$$

where:

- n is the number of landmarks
- P is the camera projection matrix
- N is the number of Blendshapes
- B_0 is the neutral face shape
- α_i is the weight associated with Blendshape i, $0 \leq \alpha_i \leq 1$
- B_i corresponds to the i-th Blendshape
- $q^{(l)}$ represents the l-th 2D landmark point in the image
- v_l represents the vertex corresponding to landmark l

Our optimization is an iterative process, where we first solve for the parameters of the shape in the Morphable Model, while keeping the expression parameters constant and then optimize for the parameters of the facial expressions, while keeping the shape parameters constant. We perform this optimization as before in MATLAB using the fmincon function for nonlinear optimization.

Note: The Blendshapes B_i in the previous step are specific to the Neutral face shape B_0 for this current iteration of the algorithm. This means that every time the neutral shape is changed by fitting the Morphable Model to the images of the actor, the corresponding Blendshapes will need to be generated from that neutral face. This is a straightforward procedure as our neutral face shape is already in correspondence with our template Blendshape model. So we make use of the Deformation Transfer Algorithm [16] from section 3.3.1 in order to do this at every iteration.

TEXTURE PARAMETERS

To integrate textures into the new 3D model, the shape must be first established. Once the shape of the model has been found as above, the texture can be extracted from the images. By projecting the shape model onto the image plane, the error at pixel level for colour can be established. Initially the average texture should be applied to the 3D model. The parameters for the 3D texture model can be optimized as with the shape model with the same bounds on parameters of ± 3 standard deviations for 99.7% of the textures to be represented. With each point projected onto the image plane, the error for each image is described as the mean of the Euclidean distance between the colour of the projected point, $[R, G, B]$, and the colour of the

point on the image, $[r,g,b]$, for all pixels within the face.

$$E_T = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(R_i - r_i)^2 + (G_i - g_i)^2 + (B_i - b_i)^2} \tag{16}$$

However, this method is light sensitive and requires consistent lighting for all the images with no highlights, causing difficulties during implementation.

Our textures for the 3D Morphable Model are obtained from the 3D scans themselves and they need to be normalized in UV-space before the model can be built using equation (8). Since our assumption is that we use a single texture value for each vertex, our texture is interpolated for values in between vertices. In our case as our mesh has 4096 vertices, we have 4096 texture samples. Figure 28 shows an example texture obtained using our scanner and the resulting interpolated texture on our deformed mesh.

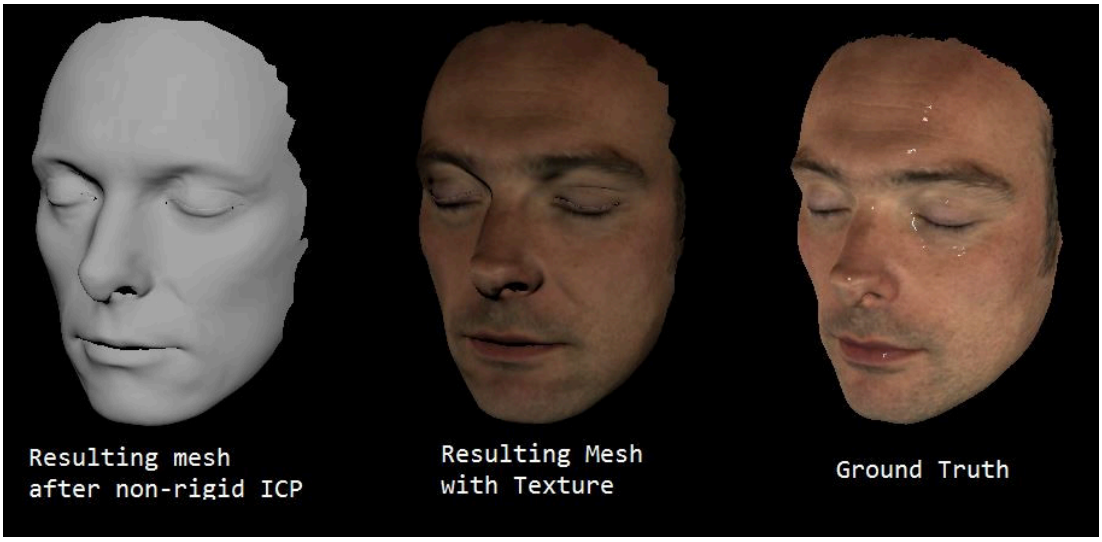


Figure 28: Texture quality obtained after interpolation vs ground-truth texture.

ADJUSTING THE COST FUNCTION TO INCLUDE WEIGHTS

The cost function outlined in equation (15) provides acceptable results as shown in Figure 29, but further improvements can be made. The method so far provides a good estimate for the internal landmarks, but when the landmarks on the outer contours of the face are included, the results are affected disproportionately. To combat this, a weighting can be applied to the internal face landmarks as follows:

$$E_L = \sum_{l=1}^{n} w_i \times \|P(B_0 + \sum_{i=1}^{N} \alpha_i B_i)^{(v_l)} - q^{(l)}\|^2 \tag{17}$$

such that for internal landmarks, $i = 1, ..., m$ and outer contour landmarks $j = m+1, ..., n$, $w_i > w_j, \forall i, j$.

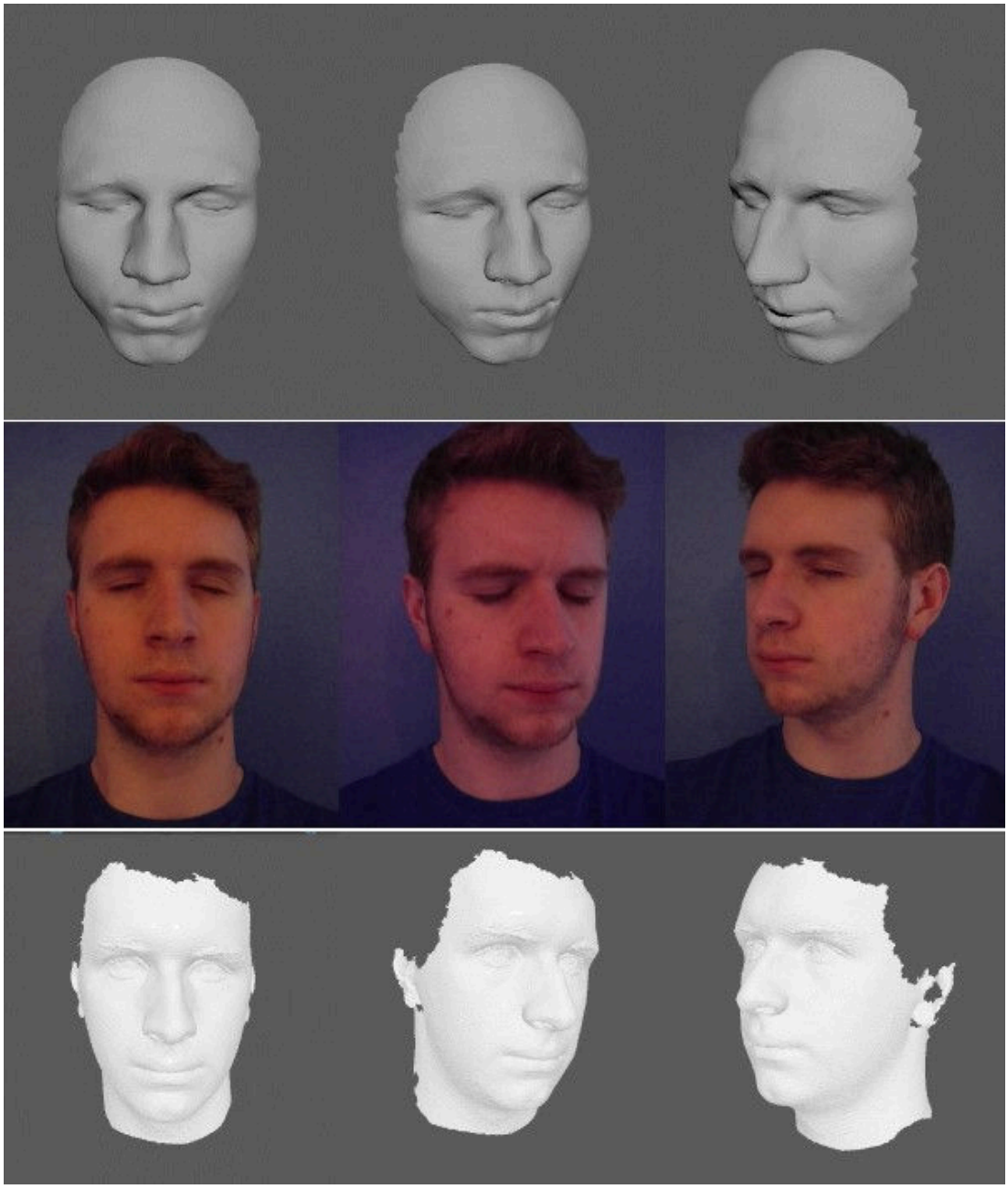


Figure 29: Results of fitting the Morphable Model parameters to images of an actor without weighting the landmarks (top row). Ground truth (bottom row).

RESULTS

Results using our weighting approach are shown in Figures 30, 31, 32, 33 and 34.

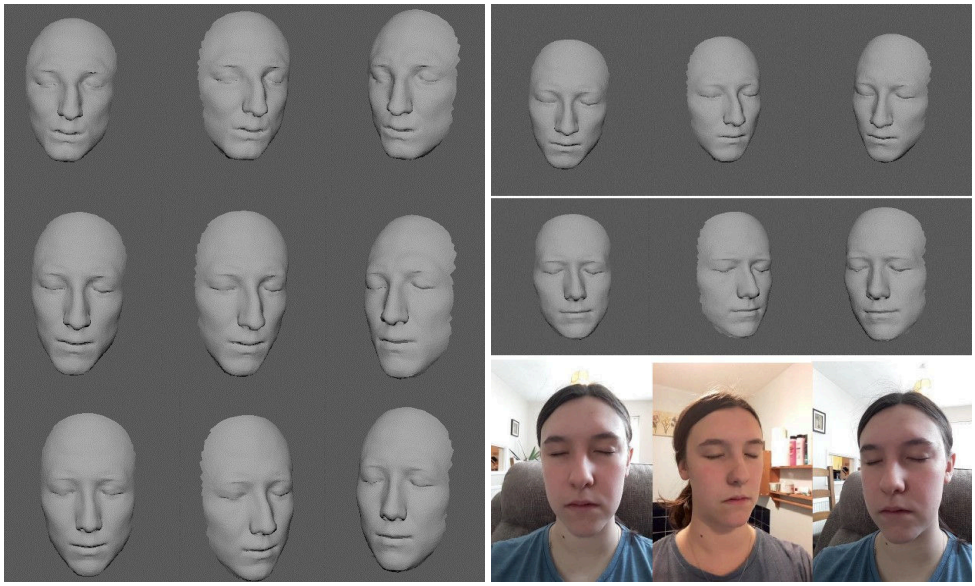


Figure 30: Results of fitting the Morphable Model parameters to images of an actor with weightings of 1, 5, 10, 15 and 20 to the internal landmarks on the face.

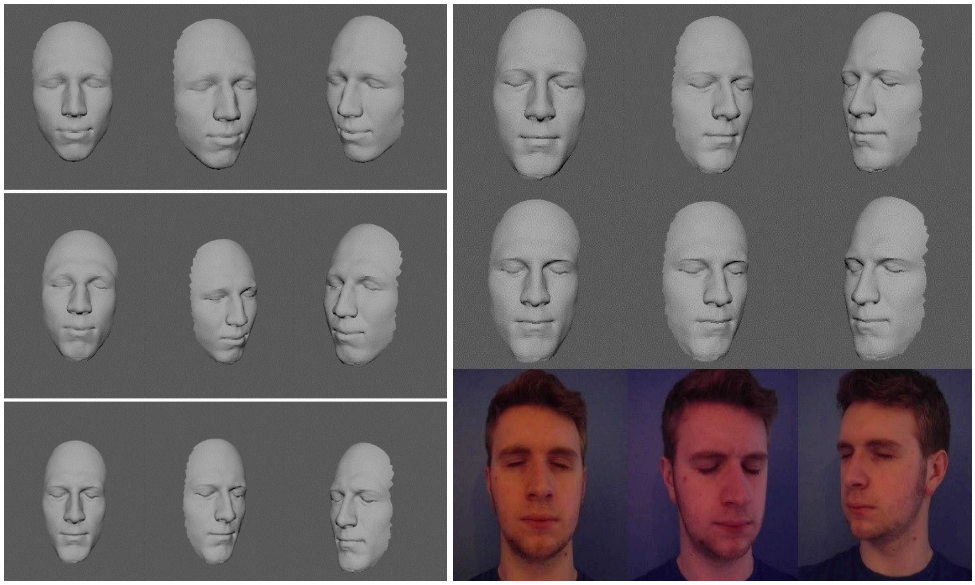


Figure 31: Results of fitting the Morphable Model parameters to images of an actor with weightings of 1, 5, 10, 15 and 20 to the internal landmarks on the face.

Using the methods described in the previous sections, we can now generate a 3D model of an actor's face, with desired topology, using just images of the user. Once we have this single

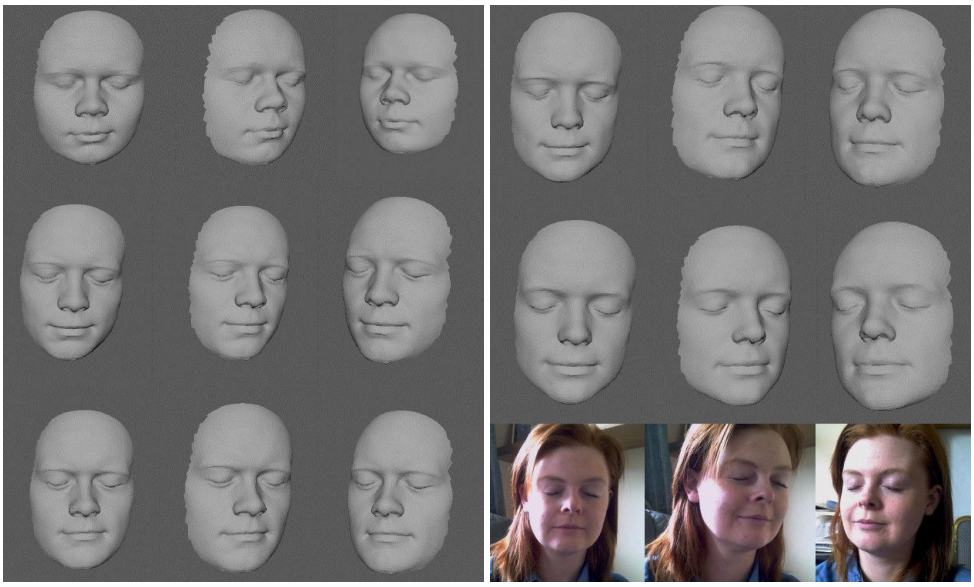


Figure 32: Results of fitting the Morphable Model parameters to images of an actor with weightings of 1, 5, 10, 15 and 20 to the internal landmarks on the face.

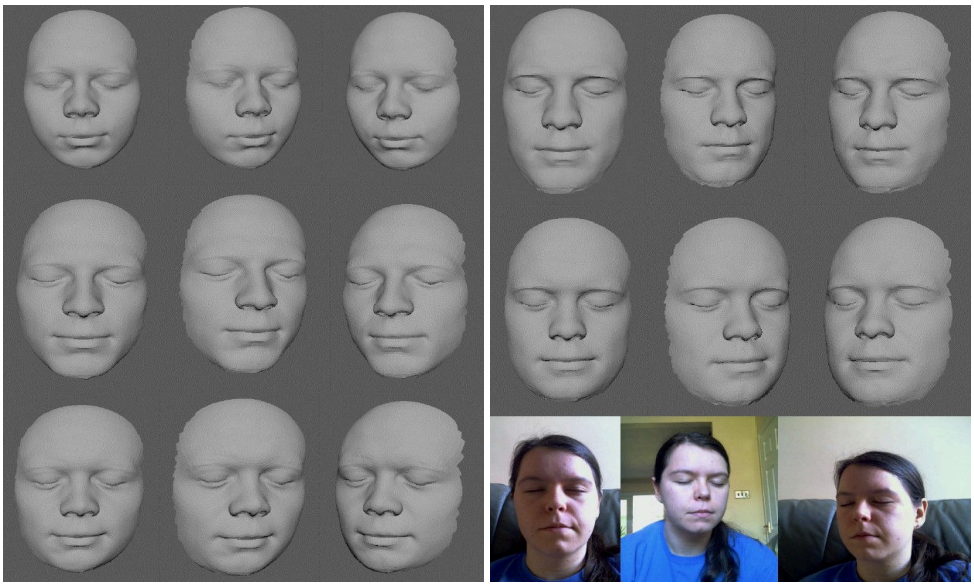


Figure 33: Results of fitting the Morphable Model parameters to images of an actor with weightings of 1, 5, 10, 15 and 20 to the internal landmarks on the face.

mesh of the actor's face, we can then proceed to the next step in the Blendshape pipeline, which is generating the Blendshapes specific to the actor. This is discussed in the next section.

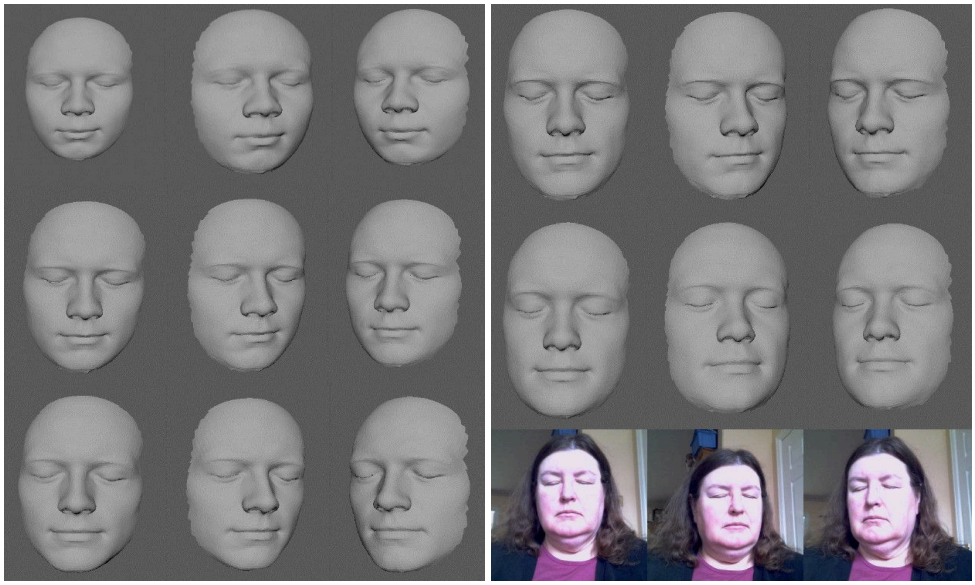


Figure 34: Results of fitting the Morphable Model parameters to images of an actor with weightings of 1, 5, 10, 15 and 20 to the internal landmarks on the face.

3.3 Automatic Generation and Personalization of Blendshapes

Obtaining the neutral expression shape of the user with desired topology is the first step in the Blendshape generation pipeline. Once we have this, it is necessary to generate Blendshapes for the actor that are specific to his facial structure. This can be done using a simple vertex offset transferred from a template Blendshape but better results can be obtained using the method of [16] who use a triangle based deformation transfer algorithm. Going further, using the method of [17], user specific nuances which are not present in the template Blendshape can be added in an organic way into the Blendshapes. This requires a few more scans (with same topology as the neutral) of the actor in different poses. We also present our algorithm to learn a statistical mapping between a person's facial structure and facial expression. Finally we discuss how our pipeline adds in high frequency details that are not present in the template Blendshape and that are unique to the user. We also discuss how we visualize and manipulate these Blendshapes.

3.3.1 Deformation transfer

Sumner and Popovic [16] in 2004, presented the Deformation Transfer algorithm. Given a source Blendshape model and a mesh of an actor's face, the objective of the Deformation Transfer algorithm is to transfer the change in shape exhibited from the source face to its individual Blendshapes, onto the target mesh of the actor's face in order to generate the corresponding Blendshape model for the actor's face. The deformation of the source face is represented as a collection of affine transformations for each triangle of the source mesh. The changes in orientation, scale and skew induced by the deformation on the triangle is encoded in the non-translational portion of the affine transformation. The three vertices of a triangle, by themselves, do not establish how the space perpendicular to the triangle deforms though and in order to resolve this, a fourth vertex is added in a direction perpendicular to the triangle.

Let v_i and $\tilde{v_i}$, $i \in 1...3$, be the undeformed and deformed vertices of the triangle respectively. The fourth undeformed vertex is computed as follows:

$$v_4 = v_1 + (v_2 - v_1) \times (v_3 - v_1) / \sqrt{|(v_2 - v_1) \times (v_3 - v_1)|} \tag{18}$$

Similarly the vertex $\tilde{v_4}$ is calculated. The cross product is scaled by the reciprocal of the square root of its length and this causes the perpendicular direction to scale proportional to the length of the triangle edges.

An affine transformation given by a 3×3 matrix Q and displacement d transforms these 4

vertices as follows:

$$Qv_i + d = \tilde{v}_i, \quad i \in 1...4. \tag{19}$$

Subtracting the first equation from the others to eliminate d, and rewriting in matrix form, we get $QV = \tilde{V}$ where:

$$V = [v_2 - v_1, v_3 - v_1, v_4 - v_1], \quad \tilde{V} = [\tilde{v_2} - \tilde{v_1}, \tilde{v_3} - \tilde{v_1}, \tilde{v_4} - \tilde{v_1}] \tag{20}$$

Q is then given by

$$Q = \tilde{V}V^{-1} \tag{21}$$

Equation (21) is then used to compute the source transformations $S_1.....,S_{|S|}$ that encode the change in shape induced by the deformation, where S refers to the set of triangle indices for the source mesh.

In order to account for the proper positioning of triangles, we cannot apply S_i directly to the corresponding target triangle since S_i does not encode the position of triangles relative to its neighbors. This can be seen in Figure 35(A). This is because the deformation representation allows too many degrees of freedom.

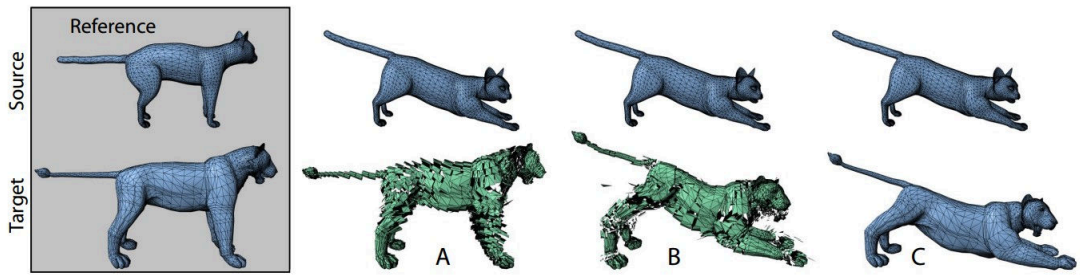


Figure 35: [16] (A) Using only the non-translational component of the source transformations transfers the change in orientation and scale to the target triangles but does not position them appropriately relative to their neighbors. (B) Using the source displacements gives a disconnected shape since consistency requirements are not enforced. (C) Deformation transfer solves a constrained optimization problem for a new set of target transformations that are as close as possible to the source transformations while enforcing the consistency requirements: shared vertices must be transformed to the same place.

In order to ensure that the affine transformations applied to neighboring triangles that share vertices are consistent with each other, we have to make sure that the shared vertices are trans-

formed to the same location.

For the set of target affine transformations $T_1 + d_1 ..... T_{|T|} + d_{|T|}$ this requirement is:

$$T_j v_i + d_j = T_k v_i + d_k, \quad \forall i, \forall j, k \in p(v_i) \tag{22}$$

where $p(v_i)$ is the set of all triangles that share vertex v_i.

In order to transfer the source deformation onto the target mesh while maintaining these consistency requirements Figure 35(C). Deformation transfer minimizes the difference between the non-translational components of the source and target transformations and enforces the consistency constraints in Equation (22) by solving the following constrained optimization problem for the target affine transformations:

$$\underset{T_1 + d_1 ..... T_{|T|} + d_{|T|}}{min} \quad \sum_{j=1}^{|M|} \| S_{s_j} - T_{t_j} \|_F^2 \tag{23}$$

subject to:
$T_j V_i + d_j = T_k v_i + d_k, \forall i, \forall j, k \in p(v_i)$.

A solution of this optimization problem defines a continuous deformation of the target mesh up to a global translation. The global translation can be defined explicitly by setting the displacement d_i for any target triangle.

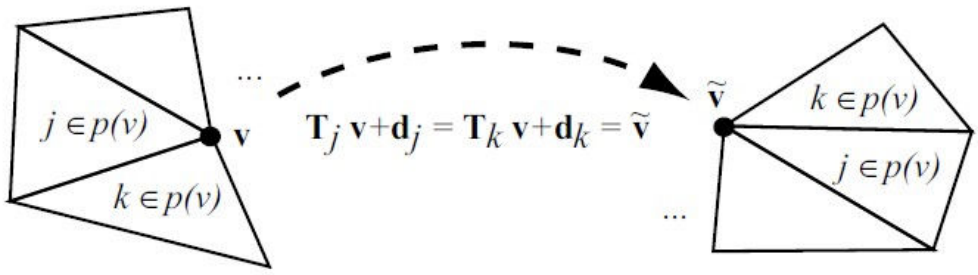


Figure 36: In order to maintain consistency, the affine transformations for all triangles $j, k \in p(v)$ that share vertex v must transform v to the same position [16].

While the formulation in Equation (23) can be solved with quadratic programming, the constraints can be eliminated by reformulating the problem in terms of vertex positions. This is achieved by defining the transformation in terms of the triangles' vertices. Rather than solving for the entries of the affine transformations, we solve directly for the deformed vertex positions.

For each target triangle, the non-translational part of the affine transformation can be written

in terms of the undeformed and deformed vertices as $T = \tilde{V}V^{-1}$. The elements of $\tilde{V}$ are the coordinates of the unknown deformed vertices.

The minimization problem is then written as:

$$\min_{\tilde{v_1}\ldots\tilde{v_n}} \quad \sum_{j=1}^{|M|} \|S_{s_j} - T_{t_j}\|_F^1 \tag{24}$$

The solution to this optimization problem is the solution to a system of linear equations.

$$\min_{\tilde{v_1}\ldots\tilde{v_n}} \quad \|c - A\tilde{x}\|_2^2 \tag{25}$$

where $\tilde{x}$ is a vector of the unknown deformed vertex locations, c is a vector containing entries from the source transformations, and A is a large, sparse matrix that relates $\tilde{x}$ to c.

RESULTS

[16] also describe how to use their method for source and target meshes that do not share the same topology. In our case though we have already ensured that the target neutral face mesh has the desired topology so applying this method is straight forward. Results from using this approach to generate Blendshapes are shown in 37, 38 and 39

Given $\mathbf{B}_0$ (from section 3.2.3), and the existing Blendshapes in the generic model we are then able to create new targets $\mathbf{B}_i$ by taking mesh $\mathbf{B}_0$ and deforming it towards the target using the deformation transfer approach of [16]. This results in a clean personalized Blendshape model for a new person, i.e.

$$\mathbf{B} = \mathbf{B}_0 + \sum_{i=1}^{N} \alpha_i(\mathbf{B}_i - \mathbf{B}_0) \tag{26}$$

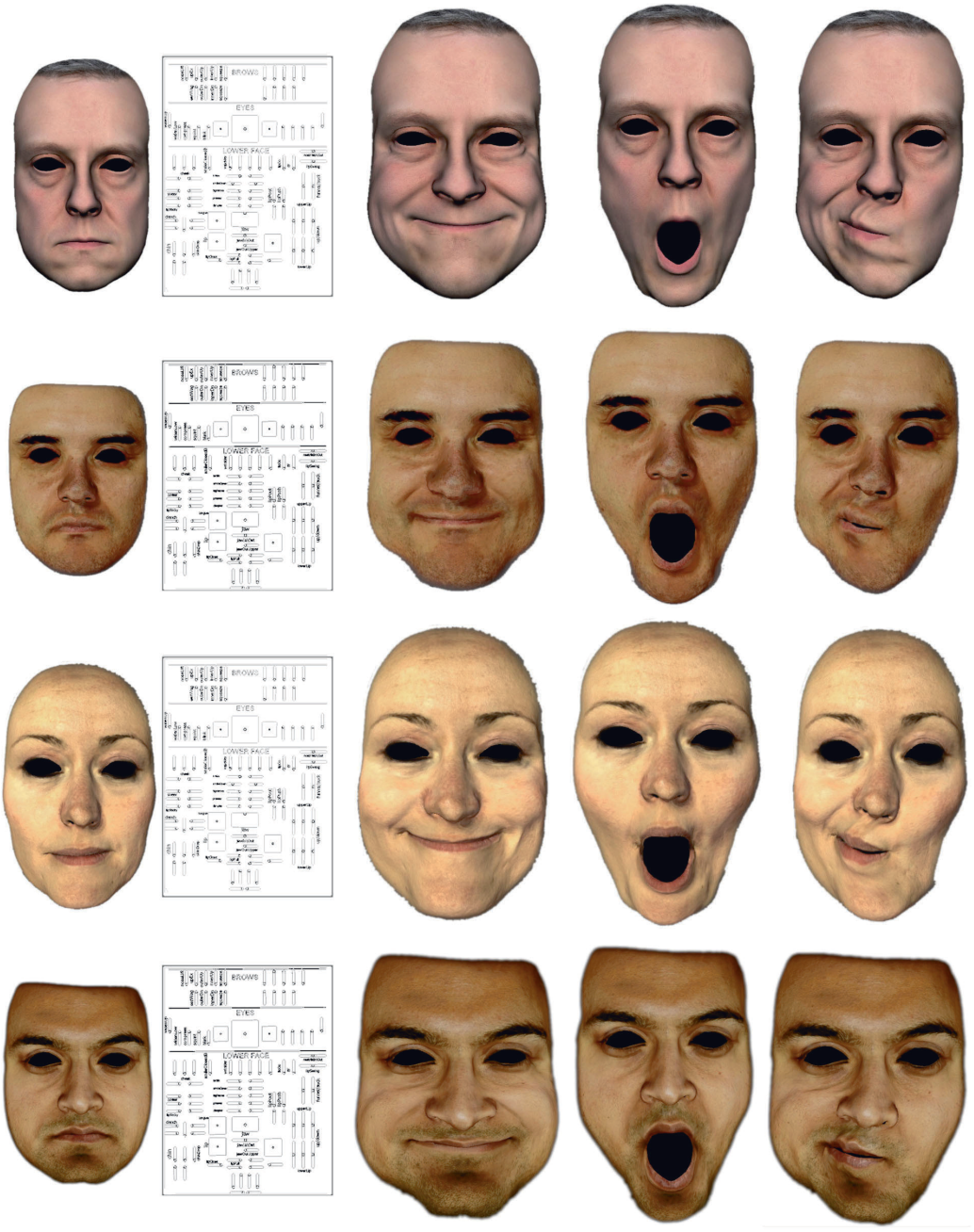


Figure 37: Results of our Blendshape generation pipeline. The top row shows the template Blendshape model. The next 3 rows show the Blendshape models generated using our method.

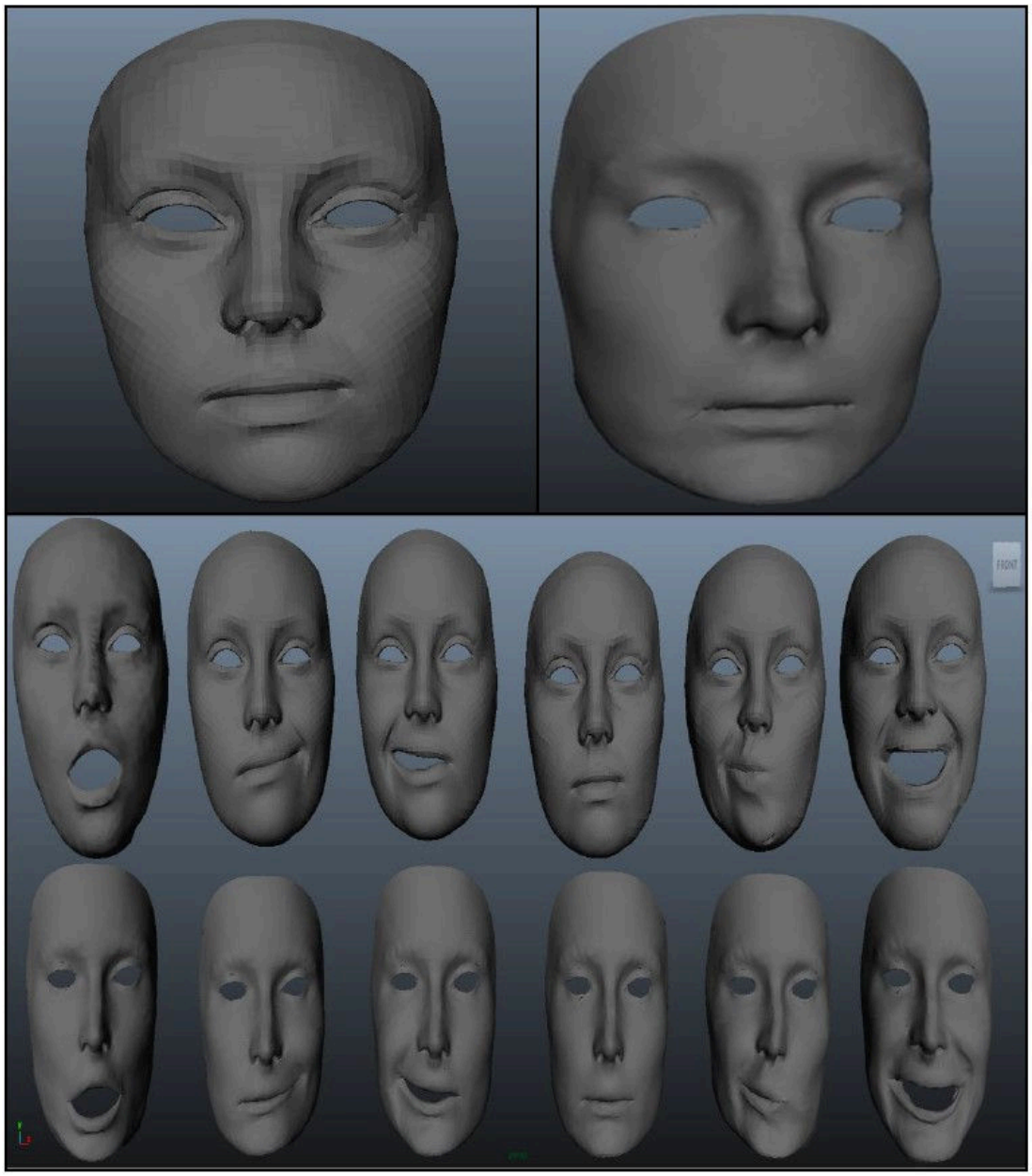


Figure 38: Results of our Blendshape generation pipeline. The top row shows the template source mesh (left) and the target face (right). The next 2 rows show the template Blendshape models and the shapes generated using our method.

3.3.2 High Frequency Details

Finally, in order to obtain the high frequency detail of an individual's face, we scan the performer in a range of additional expressions using a high resolution Artec Spider scanner [10]. We scan the performer in 5 different poses that elicit wrinkle detail around the forehead and root and side of the nose. We then add the normal maps obtained from these scanned meshes to the corresponding Blendshapes. We do this by first rigidly aligning the respective Blendshape with the high-resolution scan using manually provided correspondences between the two meshes and then generate the normal maps from the scan by casting rays from the Blendshape vertices to the high-res scan and recording the normals at the point of intersection. We do this within Maya. Figure 39 shows several Blendshapes created using this process.

Figure 39: Automatically generated Blendshapes from 3D scans of new subjects using deformation transfer with the template model as a reference. The wrinkles on the forehead (and other regions of the face), for each individual, were scanned using a high resolution scanner and added to the respective Blendshapes in a later step. Our Blendshape models contain 140 Blendshapes within an intuitive interface.

3.3.3 Example Based Facial Rigging

The Blendshapes obtained so far contain the base geometry of the actor's face — i.e. the shape of the actor's face, but it doesn't contain the actor specific nuances. This includes dimples, wrinkles and other visual subtleties. This is because the Deformation Transfer algorithm [16] only applies the transform from the template neutral mesh onto the target neutral mesh, thus only preserving the differences already present within the template Blendshape model. While subtle, these minor differences can make a lot of perceptive difference and can add to the realism of the animation, especially with regards to the identity of the actor.

In order to address this deficiency, [17] proposed an algorithm that tackles this very issue. They present a scalable design process, where the user can iteratively add more training poses to refine the Blendshape expression space. They formulate the optimization in gradient space yielding superior results as compared to a direct optimization on Blendshape vertices. Their optimization operates directly in gradient space in order to efficiently solve for Blendshapes with semantics that corresponds to those of the template rig. The Blendshape reconstruction can be edited iteratively by either adding training expressions or adapting the blending weights of the example poses. The user can specify any number of additional expressions to refine the model toward the specific geometry and motion characteristics of the user. Also in addition to the Blendshapes themselves, given an initial estimate of the weights, the algorithm also solves for the optimal weights for a given training expression.

Similar to previous sections, the neutral expression mesh and the training examples provided for this method are required to be in correspondence with respect to topology. [17] use the method of [107] in order to do this. In our approach we use the method of [11] as in previous sections. This produces a set $S = \{S_1, ..., S_m\}$ of complete meshes with connectivity of the template Blendshape model and shape of the respective scan. These are the training poses which contain the user specific nuances.

The goal is to compute a new Blendshape model $B = \{B_0, ..., B_n\}$ that better matches the geometry and the motion of the user. So in order to faithfully reproduce the training poses, we need to find Blendshapes B_i and corresponding weights α_{ij} that reproduce the training pose — i.e. $S_j \approx B_0 + \sum_{i=1}^{n} \alpha_{ij} B_i$.

The solution proceeds iteratively by solving 2 steps : The first step keeps the blending weights α_{ij} constant and modifies the Blendshapes themselves. Step 2 keeps the Blendshapes constant and solves for the optimal weights. As an initialization, the user selects appropriate blending weights on the template model that roughly correspond to each training pose S_j, giving approximate weights αij^*.

The optimization for the Blendshapes is done in gradient space similar to [16]. See 3.3.1 for details. If the actor's rest pose is given by B_0 and each of the training poses by S_j, then the gradient space representations are M_0^B and M_j^S respectively. The energy function for faithfully reproducing the training poses is given as follows:

$$E_{fit} = \|M_j^S - (M_0^B + \sum_{i=1}^{n} \alpha_{ij} M_i^B)\|_F^2 \tag{27}$$

They postulate that the deformation gradients of the user specific Blendshapes B_i and the template Blendshapes A_i should be similar. This means that $G_{B_0 \to B_0+B_i} \approx G_{A_0 \to A_0+A_i}$. We can write $G_{B_0 \to B_0+B_i} = (M_0^B + M_i^B)(M_0^B)^{-1}$ and define the regularization energy as

$$E_{reg} = \sum_{i=1}^{n} w_i \|M_i^B - M_i^{A*}\|_F^2 \tag{28}$$

where $M_i^{A*} := G_{A_0 \to A_0+A_i} \cdot M_0^B - M_0^B$, which can be computed from the template shapes and the target rest pose. The semantics of the prior Blendshape model are preserved by using the regularization weights w_i. The weights w_i are given as $w_i = ((1 + \|M_i^A\|_F)/(\kappa + \|M_i^A\|_F))^\theta$. This weighting term ensures that if the template Blendshape exhibits larger deformation, then we want the target Blendshape to reflect that and have the deformation gradients deviate more from the template prior to account for geometric differences of the two characters. Similarly, if the template shape moves little, then the same applies to the target shape. This whole term ensures that the semantics of the template Blendshape are preserved, while also capturing user specific differences.

The final energy is given as $E_A = E_{fit} + \beta E_{reg}$, where β is a weighting term. Minimizing this amounts to solving a linear system.

Given the computed set of Blendshapes B, we solve for the optimal weights α_{ij}. The user specified weights for the training poses α_{ij}^* are used as soft constraints. The energy function is given as:

$$E_B = \sum_{k=1}^{N} \|v_k^{B_0} + \sum_{i=1}^{n} \alpha_{ij} v_k^{B_i}\|_2^2 + \gamma \sum_{i=1}^{n} (\alpha_{ij} - \alpha_{ij}^*)^2 \tag{29}$$

where $v_k^{s_j}$ and $v_k^{B_i}$ are the vertices of the training pose S_j and and Blendshape B_i respectively, and N is the total number of vertices. γ is a weighting factor that balances fitting and regularization. The Blendshape weights as usual are constrained between 0 and 1. This optimization can be solved using quadratic programming.

These 2 steps are performed iteratively in order to optimize for the Blendshapes that best capture the actor's nuances while keeping the semantic structure of the template Blendshape. This results in a Blendshape set that is adapted to the actor's shape and motion characteristics on top of having the base geometry of the actor's face.

RESULTS

Figure 40 shows the input template Blendshapes that we use in our algorithm i.e. A_i. Figure 41 shows the target face neutral mesh obtained using the 3D scanner and deformed in order to have the same topology as A_0. Note that when scanning the subject, we have the eyes closed and we obtain the open-eye version of the same shape using the algorithm of [16], which we actually feed to the algorithm. We do this because we find it is easier to deal with the folding of skin around the eyes of the actor in this fashion as opposed to when the actor's eyes are open. Figure 42 shows the training poses that are obtained from the scanner and the deformed versions of these poses i.e. S_i, that are fed to the algorithm. Finally the results of applying the algorithm can be seen in Figure 43.

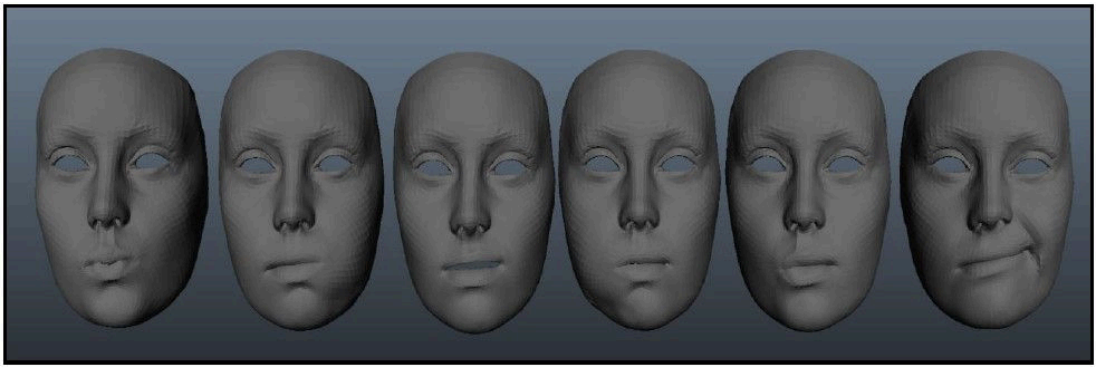


Figure 40: Source template Blendshapes used as input to our algorithm i.e. A_i.

Figure 41: Target face neutral shape obtained using scanner (left) and deformed using the algorithm of [11]. This represents B_0.

Figure 42: The training poses obtained from the scanner (Top) and the re-topologized training poses provided to the algorithm i.e. S_i (Bottom).

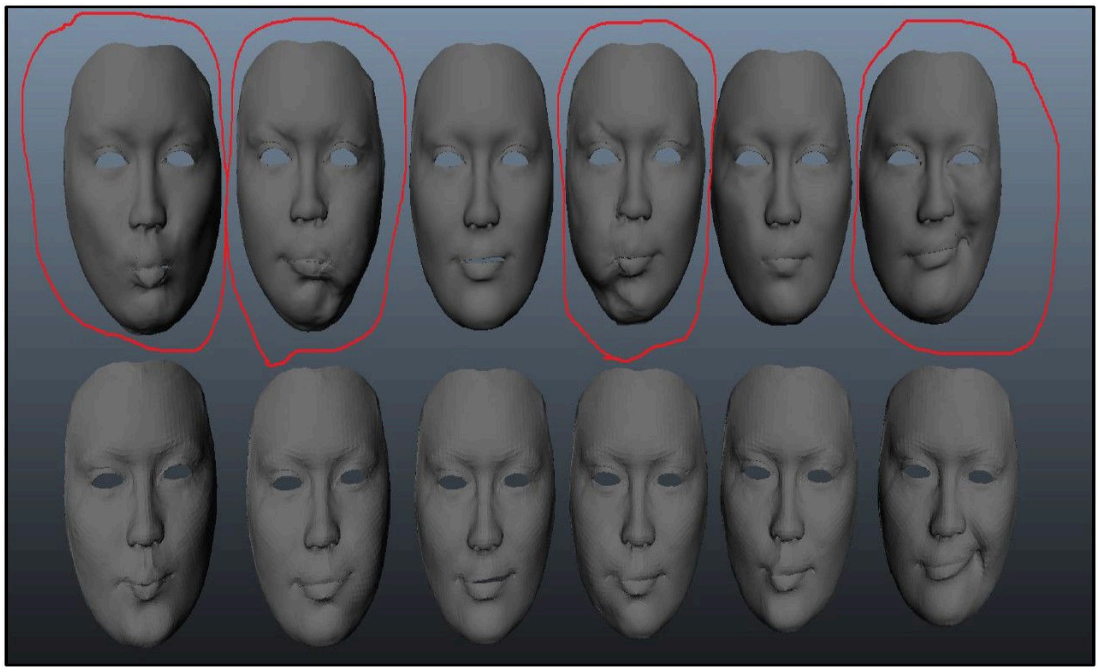


Figure 43: Results of our algorithm. The top row shows the effect of the training poses on the Blendshapes using the algorithm of [17] compared to the bottom row, which shows the application of [16] directly, without the training poses.

3.3.4 Statistical Approach to Blendshape Personalization

The use of Deformation Transfer [16] and Example Based Facial Rigging [17] is a very useful method for automatically generating Blendshapes of an actor's face. However, the personalization to the actor's nuances requires that we obtain extra training scans of the actor's face in different facial expressions. This can sometimes be difficult or impossible to obtain. Applying the generic template Blendshapes onto every actor's face, irrespective of the shape characteristics of the actor does not seem reasonable as it is expected that different structures of the underlying shape will inevitably lead to different facial expressions — e.g. a person with a smaller jaw structure will not have the same magnitude of smile as someone with a larger jaw structure. In order to address this problem, we propose an algorithm that learns a mapping between a person's neutral face shape/geometry and the facial expression that is personalized to that face. This is a statistical approach that learns this mapping from training examples of neutral-pose to expressive-pose face pairs. So given an actor's neutral face geometry, the algorithm will generate the most plausible facial expression that corresponds to that facial structure based on the mapping it has learnt from the training data.

Our hypothesis is that there is a relation between the geometrical structure of a person's face and the facial expressions that he or she makes. In order to test this hypothesis, we evaluated our method on the mapping from a neutral face to a smile Blendshape. This can be extended to other facial expressions.

Our data consisted of 27 (20 male and 7 female) scanned face pairs — in neutral expression and smile expression, under a controlled lighting environment (two diffuse lights around the subject). The data was cleaned to remove unwanted artifacts and regions behind the ear and hair. These faces (both neutral and smile) are required to be in correspondence, so as before we use the non-rigid ICP method of [11] to deform a template mesh to both the neutral and smile scans to generate faces in correspondence. Figure 46 shows examples of our scanned and registered faces. Figure 44 shows the effect of the first 5 principal components on the smile morphable model. Note that varying the principal component bases built on the smile shapes also changes the identity of the face and not just the smile shape. This has to be addressed and we do this later by refining our model.

We propose a multivariate linear regression model to learn the mapping from the principal components of the neutral face geometry to the smile geometry. We assume that the morphable model for both the neutral expression and the smile expression has been built using the methods described in the previous sections. The regression learns a mapping between the PCA representation for the neutral face and the PCA representation of the smile expression shapes.

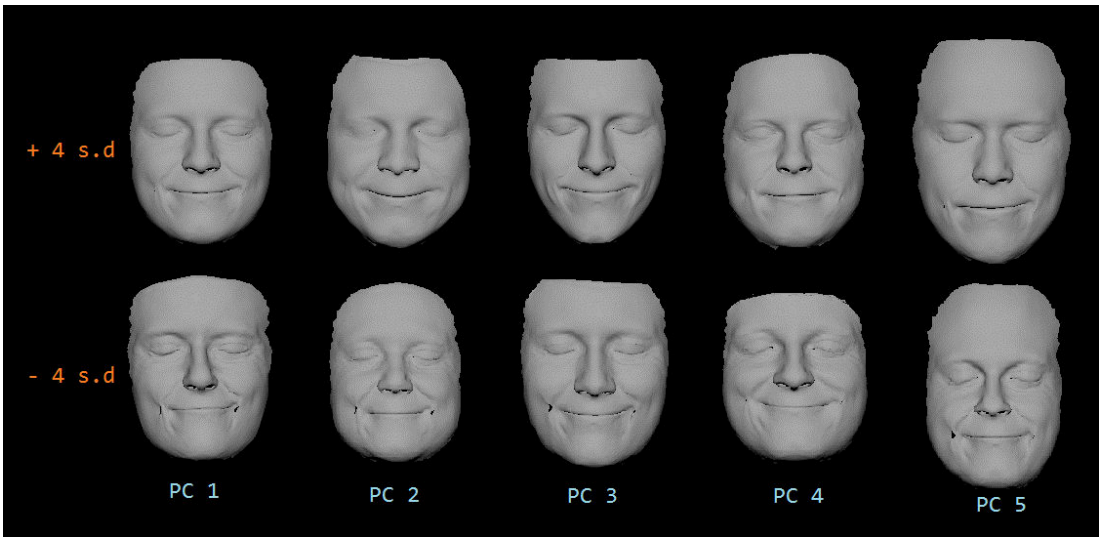


Figure 44: Effect of varying the first 5 principal component bases of our morphable model built on the smile shapes. Note how the identity of the face changes with the PCA variation. We fix this issue with our reformed model.

The input variables are in the form of a vector X given by

$$X = (N - \overline{N}) \cdot \boldsymbol{N}_{PCA} \tag{30}$$

where N is the neutral face vector containing the vertices of the neutral face shape, $N = \{x_1, y_1, z_1 ..... x_n, y_n, z_n\}$, n is the number of vertices in the mesh which is 4096 in our case, $\overline{N}$ is the mean of all the neutral face shapes and $\boldsymbol{N}_{PCA}$ represents a matrix containing the PCA bases (the eigenvectors of the covariance matrix) of the neutral face data.

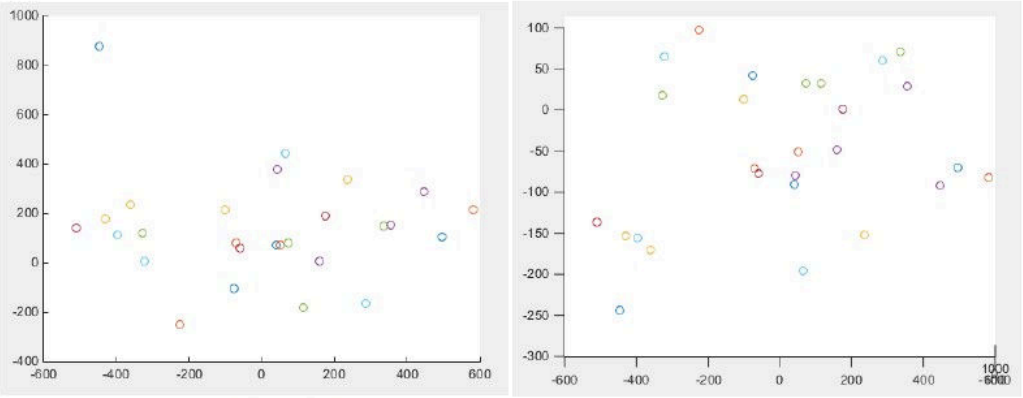
Similarly the output variables are given by

$$Y = (S - \overline{S}) \cdot \boldsymbol{S}_{PCA} \tag{31}$$

where S is the smile shape vector containing the vertices of the smile expression shape, $S = \{x_1, y_1, z_1 ..... x_n, y_n, z_n\}$, $\overline{S}$ is the mean of all the smile face shapes and $\boldsymbol{S}_{PCA}$ represents a matrix containing the PCA bases (the eigenvectors of the covariance matrix) of the smile face data.

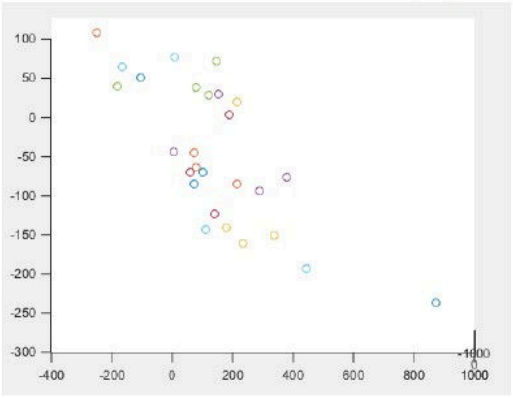
In our experiments, we use 14 principal components corresponding to the largest variance within the data for the neutral faces and we use 13 principal components corresponding to the largest variance within the data for the smile shapes.

We then train a multivariate linear regression model that maps the input vectors X to the output



(a) PC1 v PC2

(b) PC1 v PC3



(c) PC2 v PC3

Figure 45: Scatter plot of the first 3 principal components of the smile morphable model, showing the space spanned by our existing data.

Y. Given a test vector $\tilde{N}$, corresponding to a previously unseen neutral face shape, we obtain $\tilde{X}$ and predict $\tilde{Y}$ which corresponds to the coefficients of the Principal Component representation of the generated smile face. We then multiply this resulting vector by the transpose of $\boldsymbol{S}_{PCA}^{T}$ and add back the mean $\overline{S}$ in order to obtain the smile shape vector, $\tilde{S}$.

The results of this approach can be seen in Figure 47. As can be seen in the figure, while this approach can generate smile shapes that are mapped from the corresponding neutral, there is an inherent problem with the identity of the generated smile shape being changed from the input neutral shape. This is undesirable and is a consequence of training the model directly on the smile data and not on offsets of the smile shapes from their respective neutrals.

To fix this problem, we modify our Y output variables in the training data to be

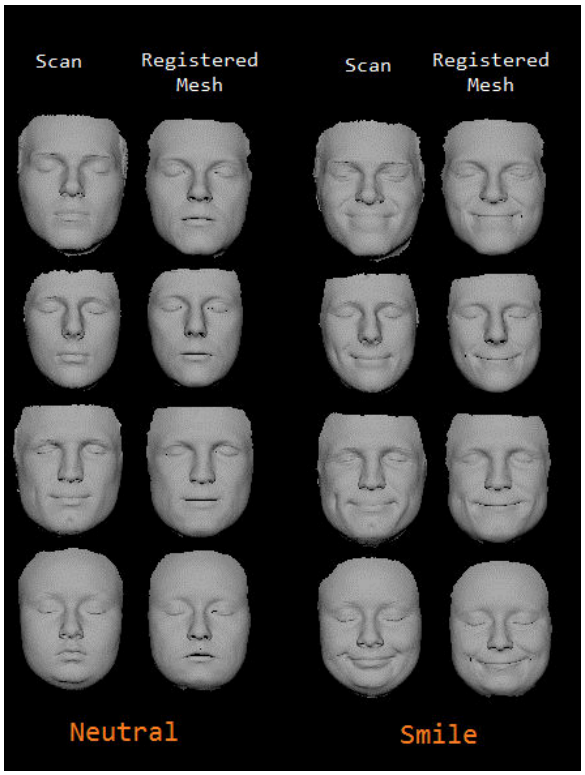


Figure 46: Examples of our scanned and registered faces for use in our mapping from neutral face to smile expression.

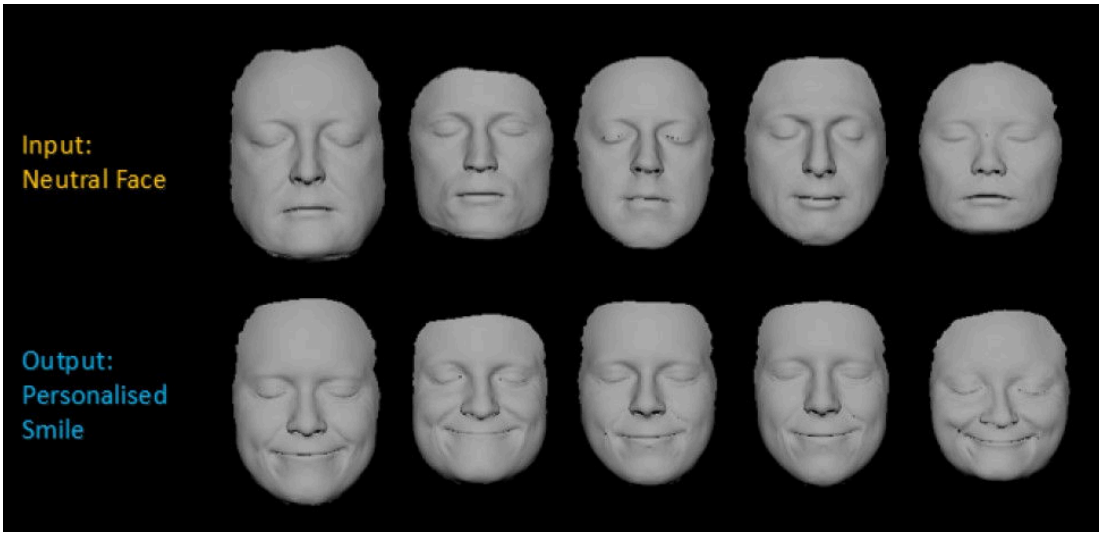


Figure 47: Results of our mapping learnt from neutral face to smile shape. As can be seen in the bottom two rows, although a mapping has been learnt, there is a change in the identity of the actor in the resulting smiles on the bottom row. This is undesirable.

$$Y = (S_\Delta - \overline{S_\Delta}) \cdot \boldsymbol{S}_{\Delta PCA} \tag{32}$$

where $S_\Delta = S - N$, i.e. the offset of the smile shape vector from its corresponding neutral shape vector, $\overline{S_\Delta}$ is the mean of all the S_Δ vectors and $\boldsymbol{S}_{\Delta PCA}$ is the PCA bases of the mean centered S_Δ vectors.

With this change, the mapping is now from the PCA representation of the neutral shape vector to the PCA representation of the smile offsets from the respective neutrals. This decouples the identity from the smiles and we only obtain the smile offsets as an output of the regression model.

The results after performing this alteration can be seen in Figure 48.

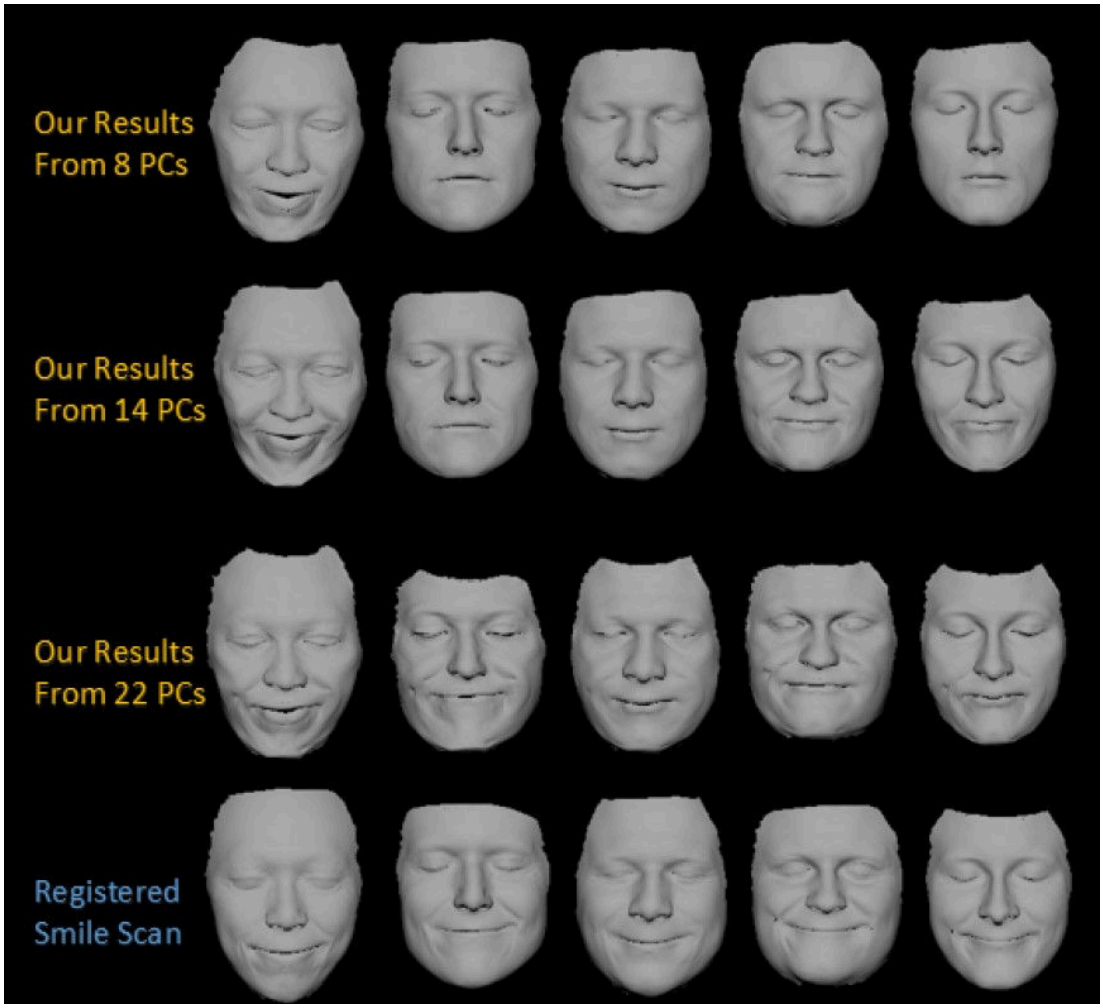


Figure 48: Results of our mapping learnt from neutral face to smile shape after adjusting for identity as in equation (32)

We compare the results obtained using our method with that obtained by using the Deformation Transfer algorithm of [16] and also to the results obtained by simply adding the average smile offset, $\overline{S_\Delta}$ to the input neutral face vector $\tilde{N}$. The results of this comparison can be seen in Figure 49.

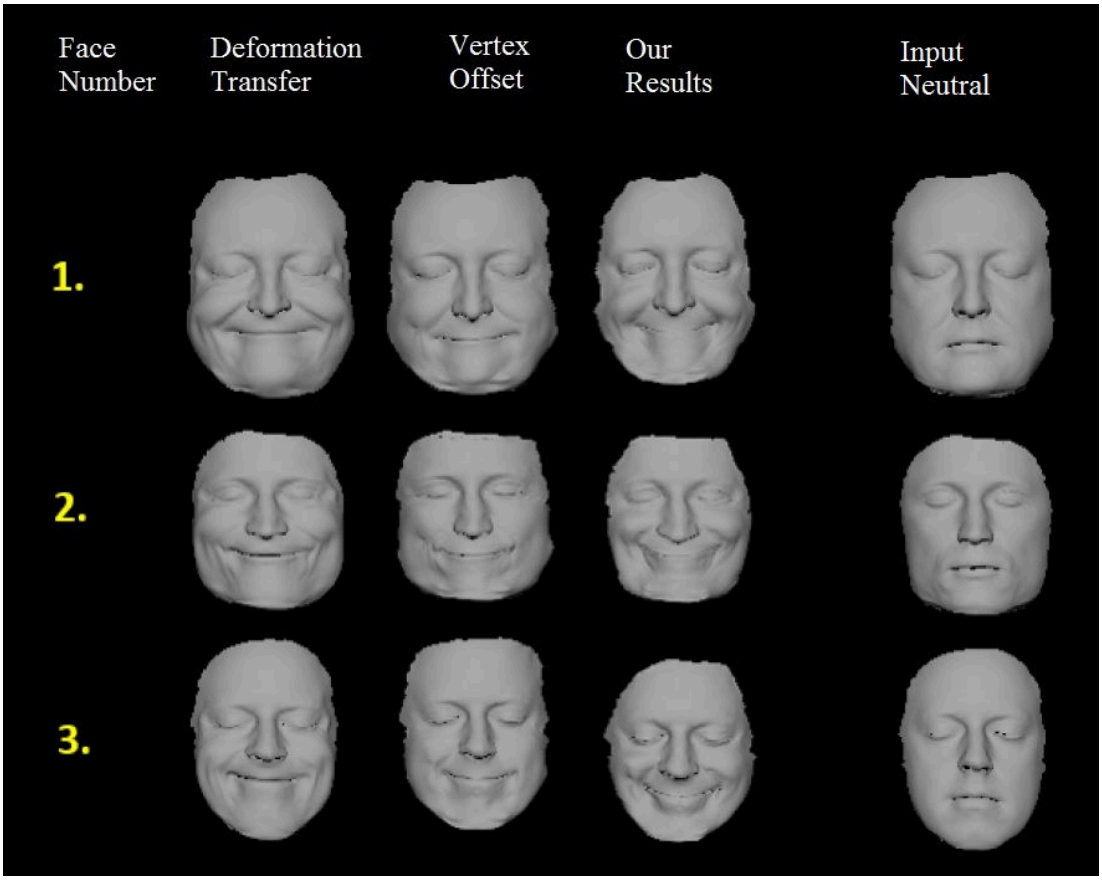


Figure 49: Our results (third column) compared to the results from applying deformation transfer [16] to the neutral face (first column), applying the average smile vertex offsets to the neutral face (second column) and the input neutral face (last column).

We also performed quantitative analysis of our results by using a sum of squares error between the generated smile expression and the ground truth expression. We first aligned the faces at the origin. By performing this test we discovered that our system's results came back as having the lowest SSE error, 72% of the time.

Smile Generation Method	Count	%	SSE
Deformation Transfer	1	4%	8.69E+06
Vertex Offset	6	24%	8.20E+06
Our Method	18	72%	3.74E+06

Table 1: Quantitative results using sum of squared errors of our generated smile shapes from the ground truth data.

3.4 Blendshape Rig and Visualization

In order to visualize our results, we make use of the 3D rendering software Maya. Our rig consists of the Blendshapes generated using our methods discussed in the previous sections but with additional sliders for convenience and manual editing. Figure 50 shows our Maya rig which forms the base template from which we export the Neutral mesh and the individual Blendshapes and use them in order to generate personalized Blendshapes for the actors ultimately resulting in a similar rig for that specific actor. This is used for the visualization and debugging of the animation once it is applied to the rig.

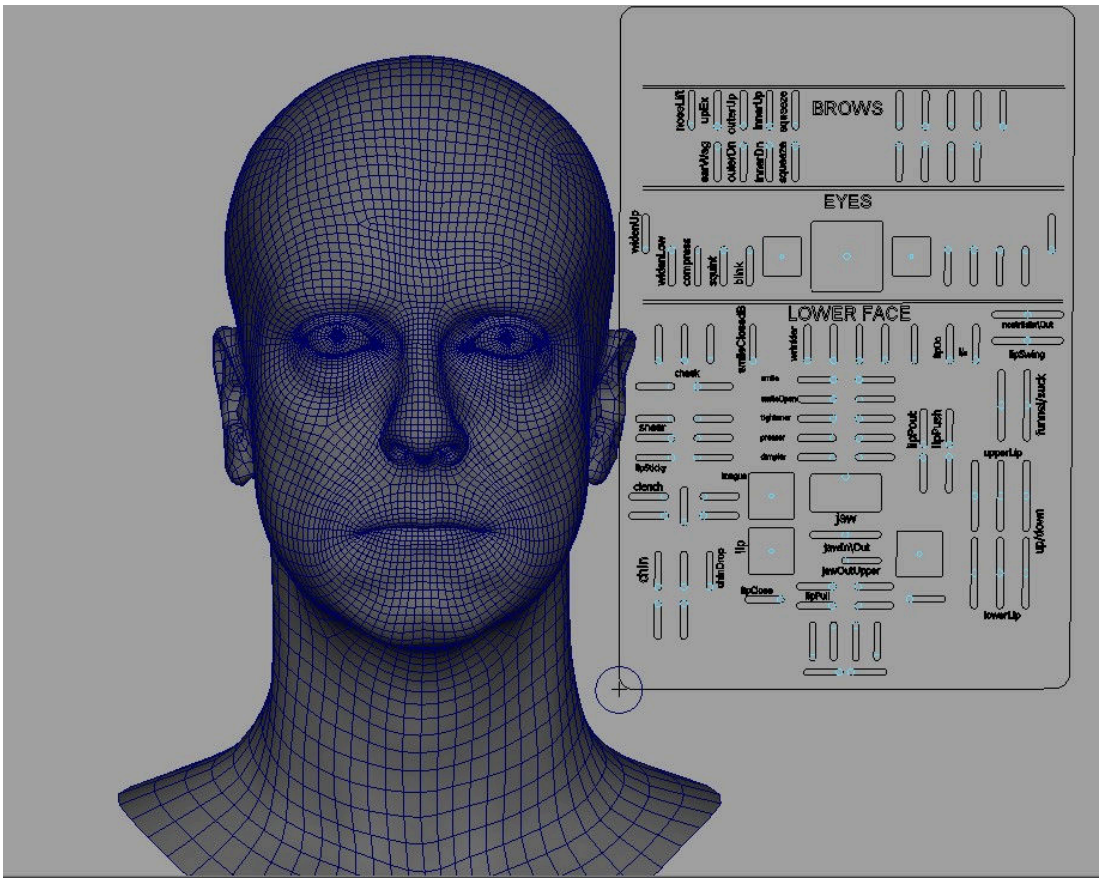


Figure 50: Example Blendshape rig in Maya which we use for visualization and debugging of our final animations. Our rig consists of 140 shapes and controllers in the form of sliders which we use both for editing and for visualization of our animation results.

While both MATLAB and OpenGL are useful for visualization, the quality required by us for final rendering is not achievable with them. In order to transfer our animation parameters within a Maya framework, we use custom python scripts which read the output of our solver (in the form of Blendshape weights) and also applies the appropriate values to the controllers

associated with each Blendshape. This allows us to not only visualize how the movement of the face is reflected on the controller-sliders but also proves invaluable in debugging errors and editing animations later on. This is very useful because the values on the sliders provide us information about the which Blendshapes affect the animation at each frame allowing us to easily detect bugs when they occur.

Over the last few sections, we discussed our Blendshape generation and personalization pipeline. We now show results of applying this pipeline to our obtained 3D scans of the actor's face. Figure 51 shows example results of complete rigs including controllers generated using our pipeline. This includes the high-frequency details added to the rig.

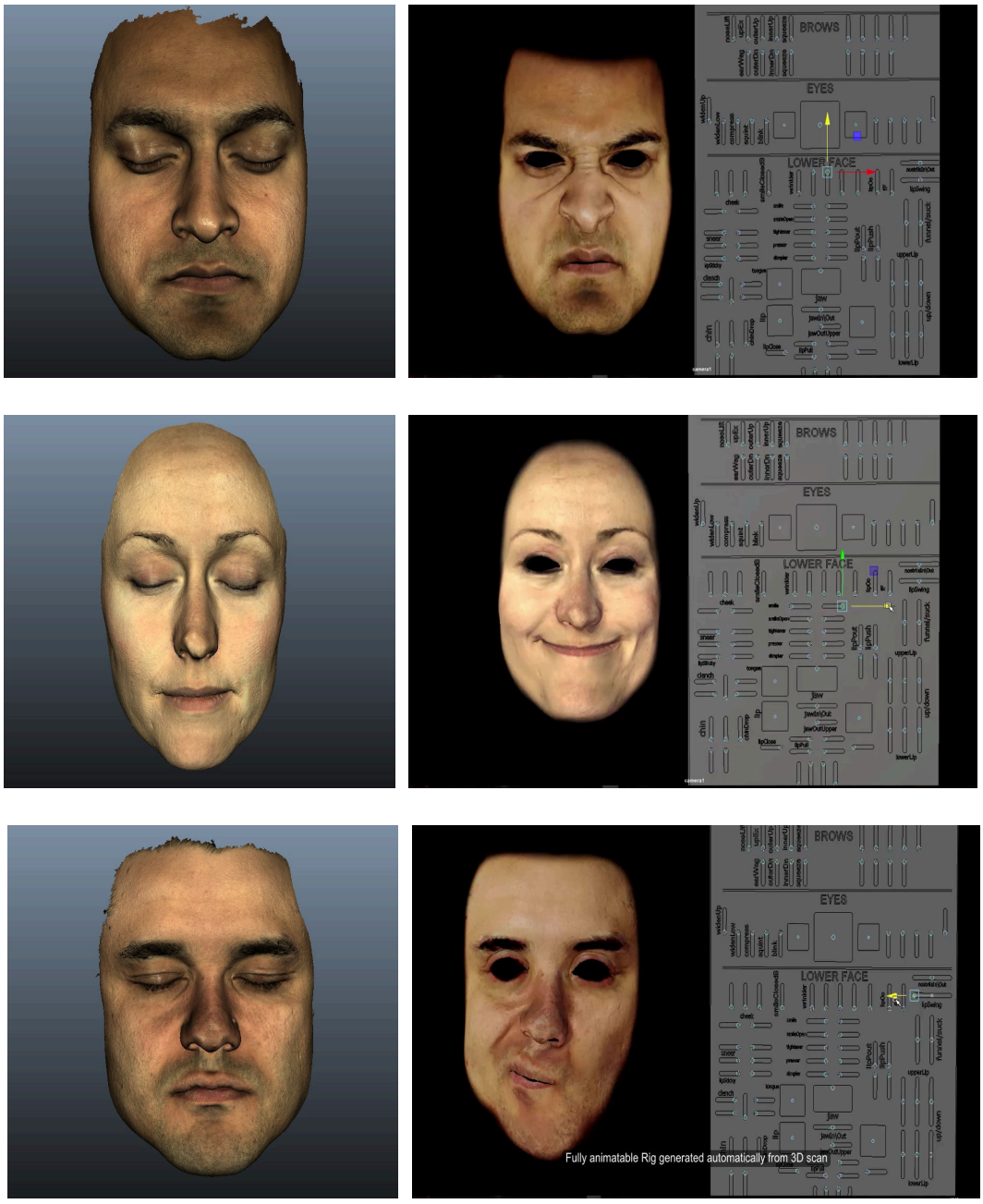


Figure 51: Example Blendshape rigs with controllers (right) generated from a single scan of the actor (left). The sliders provide direct control of Blendshapes and assist in visualizing and debugging animations The high frequency detail such as wrinkles around the top of the nose were added using the method described.

4 Capture - Tracking Facial Movements

4.1 Marker-based Capture Systems

Traditional Marker-based performance capture can be traced back to the seminal work by Williams et al. [24] in 1990. It consists of covering the face with a large number of circular dots and sometimes fluorescent markers [25] or retroreflective markers [7]. Many of the motion capture systems that have been used over the past decade are marker based systems [23]. The incident lights on the retroreflective markers is reflected back with maximum intensity back into the cameras and can be tracked efficiently. Triangulating 3D positions of these markers from the 2D inputs of the cameras allows for very accurate and relatively faster reconstructions of the markers across frames.

Marker-based tracking systems, are very popular and are used heavily in the Visual Effects community owing to their robustness, speed of acquisition and ease of use [108, 109, 48, 1, 110, 111]. Many marker based systems can be used to track marker positions in real-time giving the film director the ability to view the animations as the actor performs them, such as in ILM [23], although arguably at slightly lower fidelity compared to other denser approaches. While the use of dense motion capture systems [8, 27, 28, 26] is picking up, it is a more challenging problem to track pixels in the images, that have arbitrary surface texture as compared to reflective markers.

Marker-based systems excel at tracking and delivering sufficient motion parameters for convincing retargeting of non-human creatures or video game characters [112]. They significantly simplify the tracking process but also limit the spatial detail that can be captured owing to them being an inherently sparse feature set. Most of the motion recorded by marker-based systems have a feeling of diminished expressiveness [23] owing to the characters displaying motion that is robotic and non-human like. Although marker-based systems have their strengths in terms of accuracy and speed a lot of information in between the markers is lost. E.g. the Vicon system [7] can record dynamic facial movements at a very high temporal resolution (2000Hz) but due to the low spatial resolution (100 - 200 markers) it is not able to capture expressive facial details like wrinkles and bulges [76]. It is important to capture this information as it's perceptually very significant.

In the following sections, we discuss our marker based capture and solving pipeline and propose a method for combining markers with additional texture information in order to improve upon the traditional marker-based systems. In traditional marker-based capture, information between the marker points, such as wrinkles, bulges, complex folding of skin around the eyes

and lips which can be observed from corresponding video performances is generally overlooked. In movie production, this is resolved by animators adding this missing detail manually after motion capture and solving for animation parameters. In our work, we harness this information from the video with the use of additional sparse make-up patches between the markers. The Computer Vision community frequently uses information between sparse points to recognize facial movements and expressions, including Action Units (AUs) as described by the Facial Action Coding System (FACS) [64]. A convergence of motion capture data with such feature based approaches would therefore appear to be a promising direction in order to solve for optimal Blendshape weights, and one which is considered in our work.

There are some existing works, similar to ours, that use Blendshapes and make use of texture information on top of sparse features and also some that use all the dense texture information. We discuss these next and point out the differences with our work. Bhat et al. [39] use curves tracked along the silhouettes of the inner lips and eyelids and map these to edge contours on the mesh. They then use an arc-length based mapping to find correspondences between curve points and contour vertices in order to get a better Blendshape solve. They then also do an out-of-subspace corrective in order to improve the fit. In contrast our method automatically detects FACS poses based on the deformation of the additional patterns in order to improve the solve. Cao et al. [35] use a regression based approach that maps from UV-space to vertex displacement in order to generate high frequency detail. Their method relies on a one-time training step that uses high-resolution scans and corresponding UV-maps of different subjects in different expressions. Given an unseen actor their method can be applied directly without any pre-processing. In this respect, our method is more intrusive as we require multiple high-resolution scans of the subject in different expressions in order to obtain the high frequency data. Garrido et al. [36] densely track and use all available video information in order to improve their solve. They first track sparse 2D features accurately through the sequence and then fit the Blendshape model to this. They then compute a temporally coherent dense motion field using optical flow in order to further correct the model-to-video alignment and deform the mesh using the corrective 3D motion vector for each vertex.

4.1.1 Head Mounted System

Our marker based capture pipeline makes use of the CARA [7] head-mounted system that consists of a wearable helmet with protruding supports on which 4 RGB+infrared cameras are mounted. These cameras are focused on the actor's face and are used to track retro-reflective markers applied on the face with very high precision. The cameras are calibrated in an offline step. The helmet includes a panel consisting of a known fixed pattern of markers that is used to provide robustness in the estimation of the cameras positions and orientations relative to each other. The positions of the markers are triangulated over the performance in a post-processing step resulting in 3D locations of each marker with high fidelity and high temporal resolution.

Figure 52: The CARA head-mounted system used by our pipeline for tracking markers on the actor's face [7]. It consists of 4 cameras that are individually calibrated and track reflective markers placed on the actor up to 60FPS.

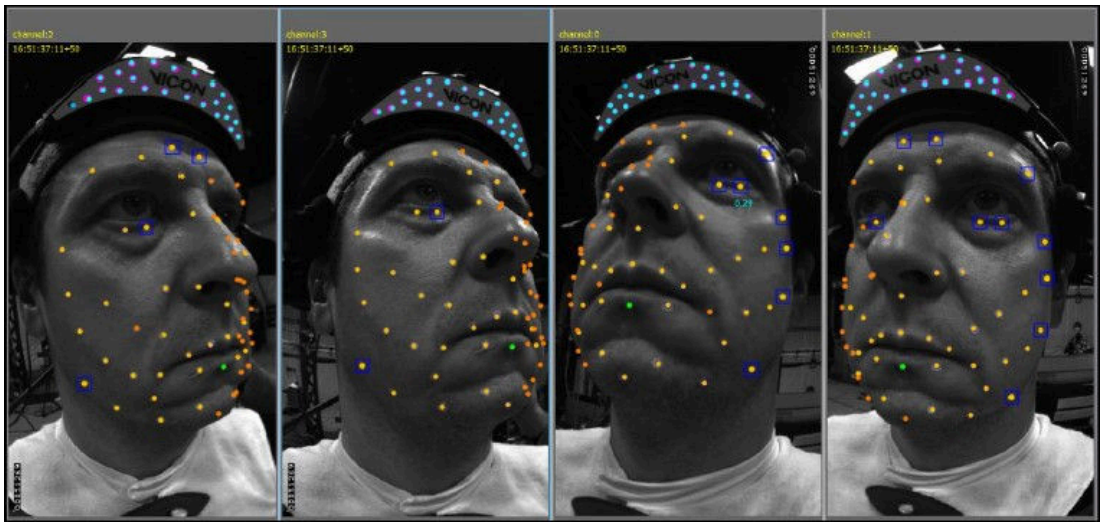


Figure 53: Images showing the Cara system in action. The system consists of a head mounted device with 4 cameras. The markers on the subjects face are tracked in all 4 cameras and the system outputs the 3D coordinates of the markers every frame of the sequence.

4.1.2 3D Marker and Mesh Registration

Before we can actually go ahead and solve for Blendshape weights, one of the important steps in Performance Based Facial capture is the 'Stabilization' of the face rig — i.e. getting the face model into the correct co-ordinate frame and making sure that the correspondence between the vertices on the mesh and the sparse markers (2D or 3D) is accurate. While this might be less of an issue in dense capture methods [5, 8] it is still a factor that needs to be addressed. If there is an error or mismatch between the location of the markers on the 3D face mesh compared to the location of the sparse landmark on the actor's face, this will lead to severe perceptual distortions in the final animation. Even subtle deviations e.g. the mouth opening more than it should owing to an error in the location of the lip-corner marker, will lead to a completely different perception of the facial expression. With this goal in mind, we present our algorithm for ensuring optimal registration of the 3D landmarks on the mesh, with the sparse markers.

Our goal is getting the 3D motion capture data $\mathbf{v}^S$ (the first frame of the performance which we assume to be a neutral pose without loss of generality) and points in $\mathbf{B}_0$, the neutral face Blendshape, in the same space, so they correspond properly. More specifically, we need to select points $\mathbf{v}^B$ corresponding to $\mathbf{v}^S$, from $\mathbf{B}_0$.

The most straightfoward approach is to manually select vertices $\mathbf{v}^B$ from the neutral Blendshape $\mathbf{B}_0$ that correspond to the facial marker placement. However, if the selection does not exactly match the facial marker placements, which is the likeliest scenario given the discrete nature of the vertices in the Blendshape model, then the solver will be influenced. This is because the optimizer will select Blendshapes to minimize this difference, adding weights to expressions which are the result of alignment error as opposed to expressions occurring on the performer's face. This is particularly problematic when the Blendshape model is low resolution, as the likelihood of vertices in the model exactly matching the placement of facial markers is lower. One minor improvement that can be done after the manual selection is a rigid-alignment between the corresponding marker-vertex points followed by a K-Nearest neighbor search for the closest vertex on the mesh to the markers. An example of this can be seen in Figure 54 We propose to determine the optimal selection of $\mathbf{v}^B$ with respect to minimizing the overall animation error (see Figure 55).

Similar to existing approaches [51], we begin with manual selection of n landmark points $\mathbf{v}^{B'}$ on the Blendshape model that are deemed to be at physically similar positions to the n 3D face markers $\mathbf{v}^S$. Using these points, Procrustes analysis is performed to estimate a rigid rotation, translation and scale between the Blendshape model and the 3D marker points of the neutral expression.

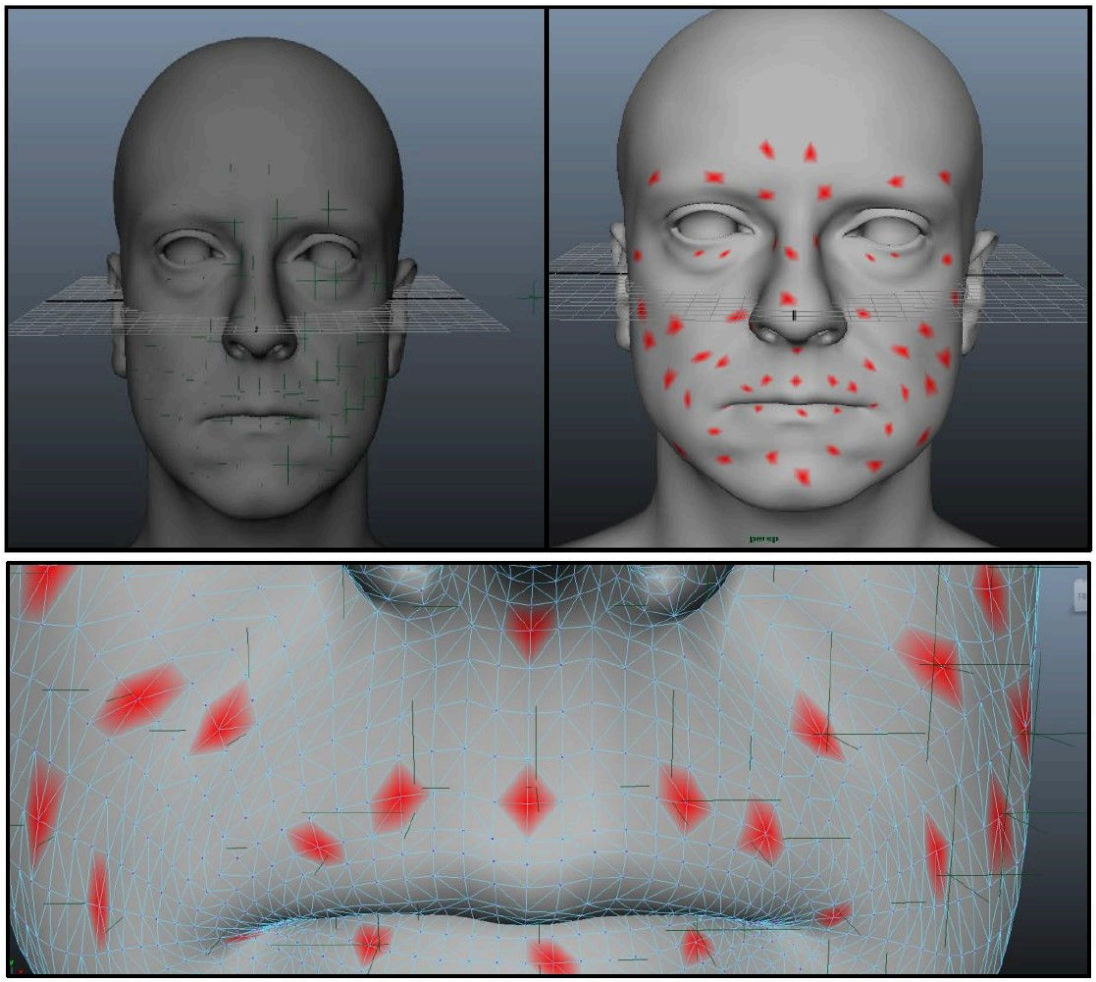


Figure 54: Initial stabilization for getting the 3D face and the Cara markers in the same space. (Top Left): Image showing the position of the Cara markers, indicated by the green crosses, with respect to the face. The markers are adjusted manually using the image of the subject as a reference. (Top Right): The mesh with the vertices corresponding to each Cara marker, shown by a red spot. The vertices are obtained using a KNN search between the markers and vertices. (Bottom):Image showing the error in the rest position. The crosses are the Cara marker locations and the nearest corresponding vertices on the mesh are shown in red. As seen there is significant error on the rest pose and there is much scope for improvement.

The initial rigid alignment will have errors, i.e. $\mathbf{v}^{B'} \neq \mathbf{v}^{S}$. Our aim is to reduce this error by finding the optimal placement of $\mathbf{v}^{B'}$. In order to do this, we ask the performer to do a Range of Motion (ROM) performance and iteratively solve for the best $\mathbf{v}^{B'}$ using the following approach:

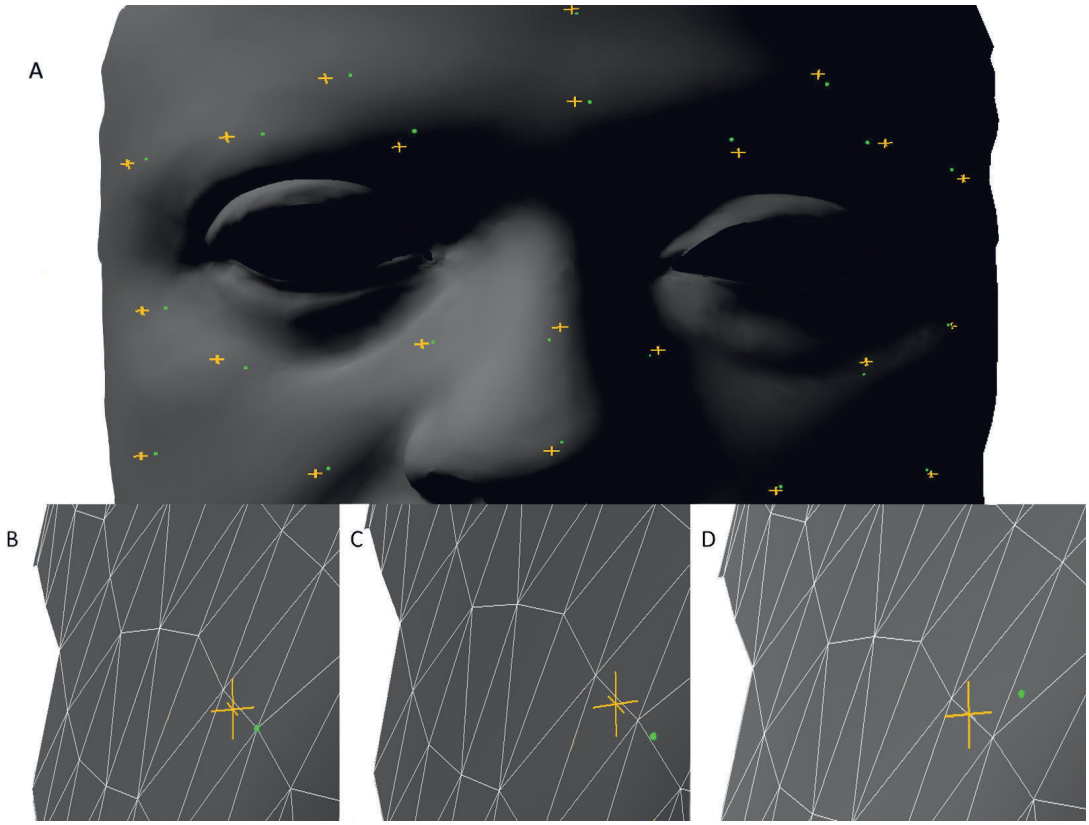


Figure 55: Barycentric optimization: **(a)** Rigid alignment between the Blendshape model (green dots) and the 3D motion capture (crosses), **(b)** closest model vertex (green) to the 3D point (cross) after rigid alignment, **(c)** new position (green) after Barycentric projection, **(d)** final position (green) after optimization over all frames of a Range of Motion (ROM) capture of the performer.

Algorithm 4 Optimal Registration of markers and vertices

1: For each marker $\mathbf{v}_i^S$ find the nearest triangle $\mathbf{T}_i$ in $\mathbf{B}_0$.
2: Estimate the Barycentric projection of $\mathbf{v}_i^S$ on $\mathbf{T}_i$, and set the corresponding position of $\mathbf{v}_i^{B'}$ to that value.
3: Using equation(33) (explained in Sec. 4.1.8), solve for the Blendshape weights over the entire ROM, giving a resultant performance $\mathbf{P}^j$.
4: For each marker $\mathbf{v}_i^S$, consider the nearest Q triangles in $\mathbf{B}_0$, estimate its Barycentric projection on these Q triangles, and re-estimate its total error over all frames across the ROM performance $\mathbf{P}^j$ for each of these triangles. The value Q should be chosen based on the mesh triangle-density and our tolerance for how much the marker position can drift from the manually selected initialization.
5: For each marker $\mathbf{v}_i^S$, let $\mathbf{T}_i$ be the triangle which gives the lowest overall marker error $\mathbf{M}_i$ over all frames.
6: Repeat (2-6) until $\mathbf{M}_i$ converges.

Finally we set $\mathbf{v}^B = \mathbf{v}^{B'}$, to be the optimal marker locations on the mesh after convergence. The iterative process terminates when the error per marker over the entire ROM sequence ($\mathbf{M}_i$) converges i.e. the change is lower than a threshold. We use a threshold value of 0.1. In our experiments, this typically requires 3-4 iterations. The intuition here is that the marker position which gives least error over the entire ROM will better capture the variation in motion of the performer even in *extreme* poses without inducing a very large error and influencing the solve. Finally, before the solve, we make sure to subtract the error in initialization ($\mathbf{M}_i$), from the target positions of the respective markers every frame. This ensures that our solver isn't influenced by the error in initialization, but is only affected by the movement of the markers.

4.1.3 Solving for Blendshape Weights

Our basic marker-based pipeline, built upon traditional approaches and without any modifications is outlined at a high level in Figure 56. The pipeline initially creates a high quality Blendshape model from a 3D scan of a performer in a neutral expression and uses 5 more high resolution scans of the performer in different expressions for acquiring the high frequency data as described in Section 3. These may also be acquired using a range of off-the-shelf or bespoke approaches [113, 106]. In our work, we use a combination of two commercial systems to acquire high quality 3D scans – Artec Eva and Spider scanners [10]. The former provides medium scale facial detail, while the latter provides small scale details such as fine wrinkles. We next use a commercial head mounted facial motion capture system – Vicon Cara [7] – to acquire facial performances of the same person. This results in 3D motion capture data (50-100 marker locations) as well as 4 video streams of the performer from the respective Head Mounted Cameras (HMC). The Blendshape model is registered to the neutral expression frame

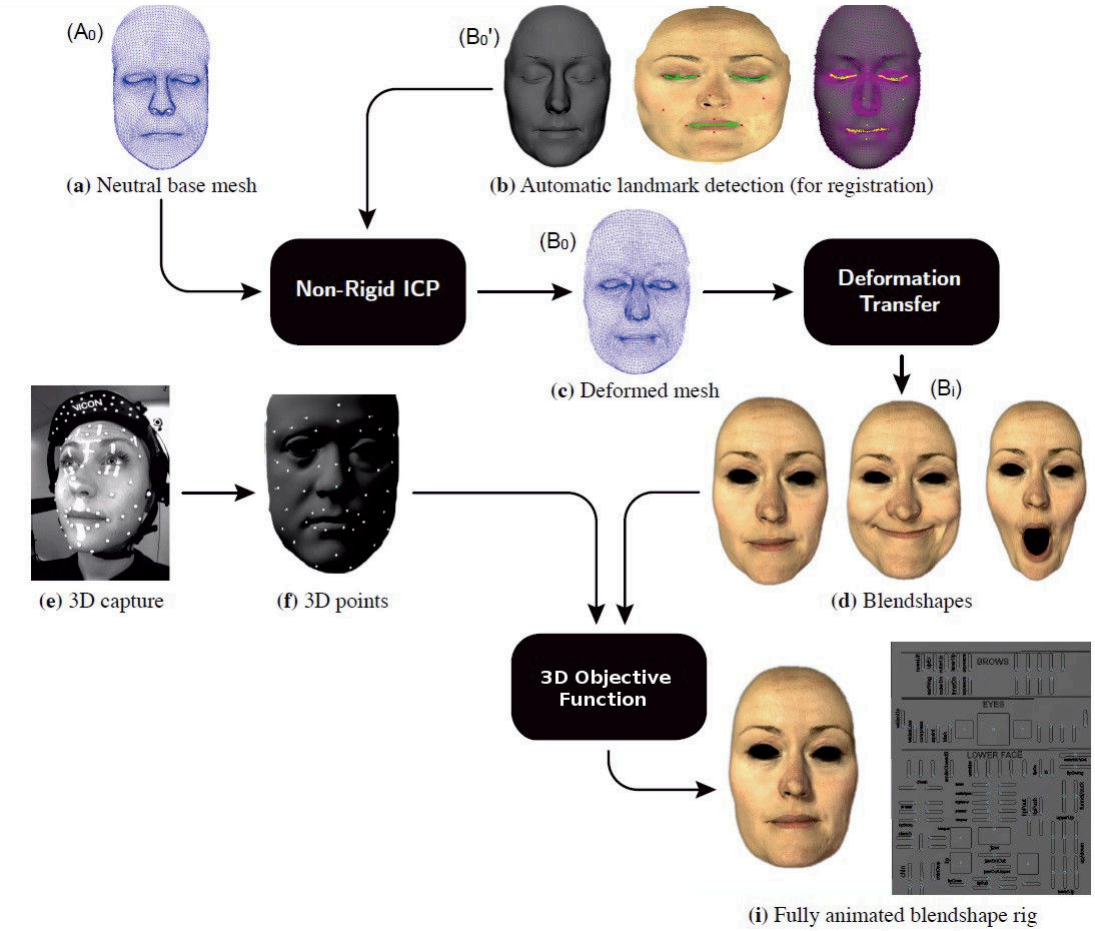


Figure 56: 3D Marker-based Pipeline Overview: **(a)** Given a template Blendshape model's neutral mesh we non-rigidly deform it to **(b)** a new 3D scan of a face, using an automatic landmark detection algorithm to assist with the non-rigid deformation, in order to obtain **(c)** the deformed mesh of the new face. **(d)** We then create a personalized Blendshape model using deformation transfer. **(e)** Given a new HMC performance, **(f)** 3D motion capture data is fed to our 3D objective function optimization to **(i)** produce a high quality Blendshape animation.

of the performance and then optimally solved for the remainder. The solver uses the 3D points to determine the optimal Blendshape combination. Given a solved performance, this is easily retargeted to new faces.

Given a personalized Blendshape model rigidly aligned to the neutral pose markers, the next task is to fit this model to the performance by optimizing the parameters of the model. As described previously, the most popular approach for achieving this reliably, especially in production, is using 3D marker positions derived using HMCs.

4.1.4 3D Objective Function

We now present our 3D objective function and explain the factors affecting the quality of the solve and how we can control them. Our objective function follows the recent trend in solving Blendshapes[17, 37]. Given a Blendshape model, with N Blendshapes and n markers on each, our core objective function is

$$E_{3D} = \arg\min_{\alpha} \|B_0 + B\alpha\ - T\|_2^2 + \beta\|\alpha\|_1 + \alpha^T\Gamma\alpha \tag{33}$$

where:

- B_0 is a $3n \times 1$ vector representing the neutral face.
- B is a matrix of size $3n \times N$, that contains the deltas for each of the Blendshapes $B_{i...N}$. In order to ensure high quality and stable solutions, the rank of the B matrix should ideally be greater than or equal to N. This depends on the number of markers, the location of these markers on the face and the Blendshape set that we use.
- α is a $N \times 1$ vector of weights with the constraint $0 \leq \alpha_i \leq 1$.
- T is a $3n \times 1$ vector representing the target markers.
- The term $\|\alpha\|_1$ is an L1-norm on α that penalizes the sum of weights. This term adds a sparsity constraint to the solver that forces the solver to choose as few Blendshapes as it possibly can to solve for the weights. It also prevents the solver from choosing opposing shapes which would cancel each other out. A sparse solution is very useful as it makes it easier for an animator to later modify the animations.
- β is a weighting factor on the L1 regularizer. The value of β should be chosen such that the term $\beta\|\alpha\|_1$ is of the same order of magnitude as the sum of squares of marker errors; too high a value of β will suppress the weights leading to muted animations.
- The Γ term is a Tikhonov regularizer that ensures that the function is convex and has a unique global solution. $\Gamma = \varepsilon I$ where ε is a very small constant and I is the identity matrix of size $N \times N$.

In the next section, we show results obtained using the above objective function applied to 3D marker inputs.

4.1.5 Results - Using Only Marker Information

We now show some results and metrics to show the performance of our 3D Objective Function. Figure 57 show the result of using our 3D objective function to solve for Blendshape weights using motion capture data using the head mounted system discussed previously. The images to the right show the resulting animations applied to the Blendshape Rig along with the activation of the controllers corresponding to the individual Blendshapes for that frame of animation. Note that this solve is done *after* doing the rigid registration step from Section 4.1.2 using Algorithm 4.

Figure 58 shows the plot of the Root Mean Square error distance between all the markers and corresponding vertices on the mesh (y-axis) for every frame of the animation.

Figure 59 shows a plot of the frontal view and the side view of the 3D markers (red cross) and the corresponding vertex position (blue cross) and the error between them, for a single frame of the animation.

Figure 60 (Top) shows the plot of the curve for the sum of all Blendshape weights per frame of the animation. Figure 60 (Bottom) shows an example curve for a single Blendshape over the course of the animation. Note that both curves are continuous and smooth. Each Blendshape should ideally have a smooth weight curve over the course of the animation. This is essential as it indicates that the resulting animation is smooth and free of jitter. As a consequence of this, the curve for the sum of the Blendshape weights should also be a smooth curve.

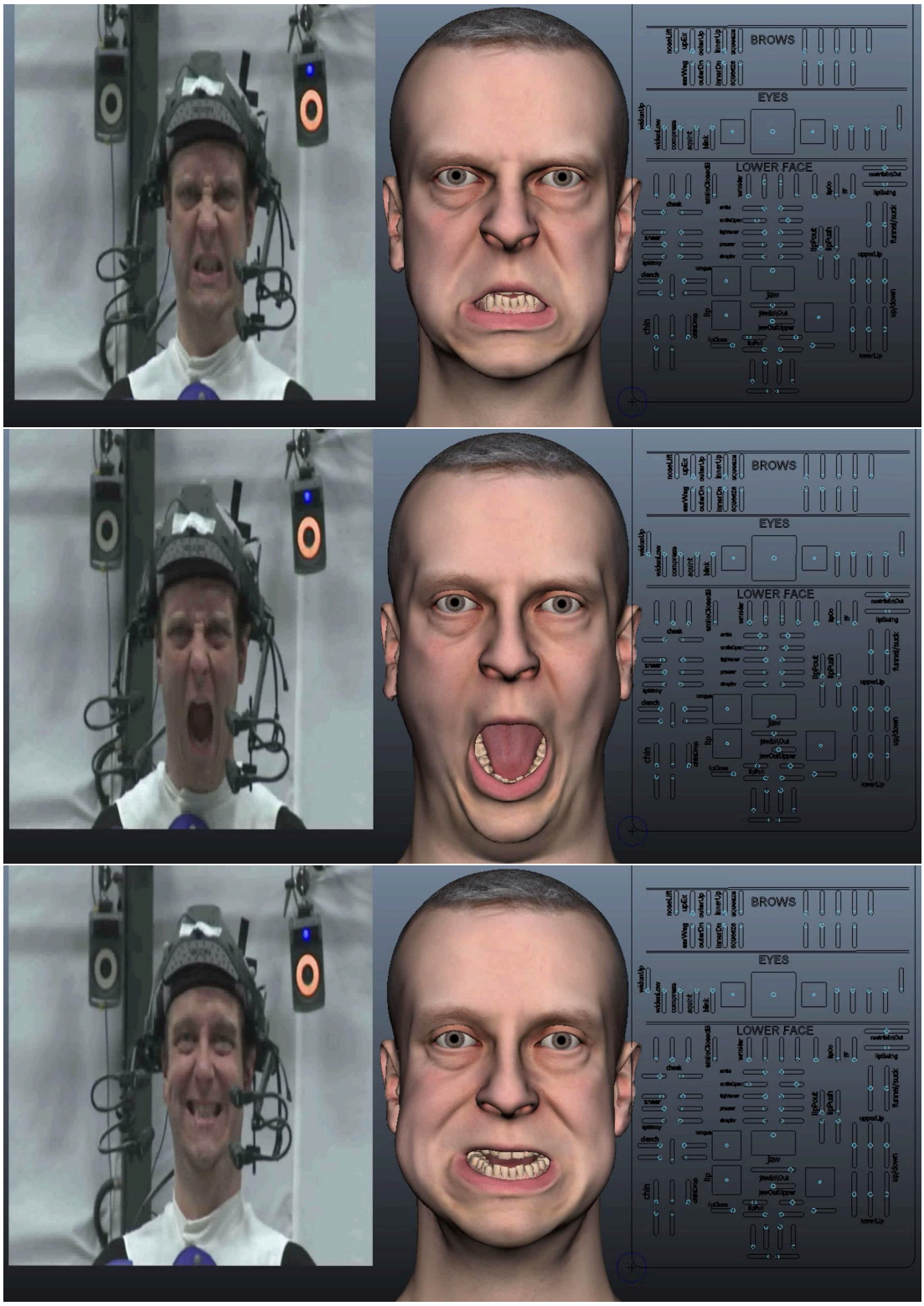


Figure 57: The result of using our 3D objective function to solve for Blendshape weights using motion capture data from the head mounted system. The images to the right show the resulting animations applied to the Blendshape rig along with the activation of the controllers corresponding to the individual Blendshapes for that frame of animation.

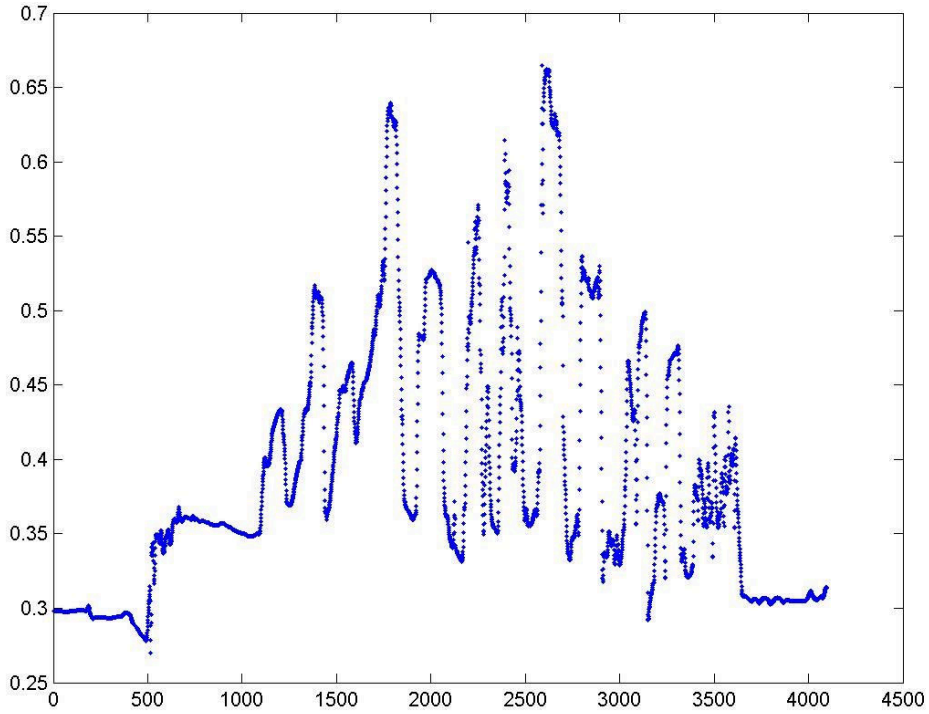


Figure 58: Plot showing the curve of the Root Mean Square error distance between all the markers and corresponding vertices on the mesh (y-axis) for every frame of the animation. The regions of high error correspond to facial movements where our Blendshape model struggles to accurately match the motion capture data. Low error values indicate that our model is able to accurately span the movement space of the actor.

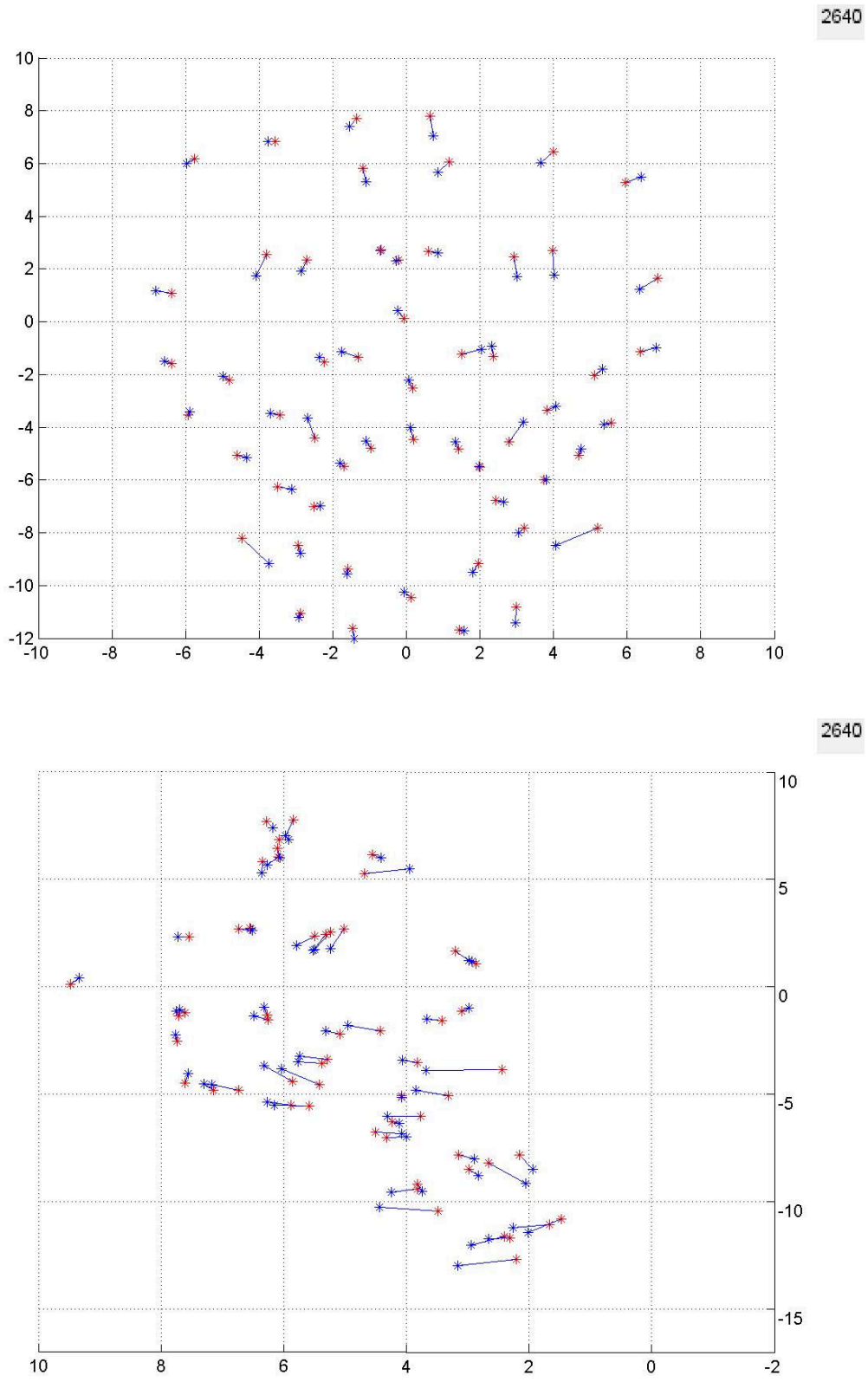


Figure 59: A plot of the frontal view (Top) and the side view (Bottom) of the 3D marker locations (red cross) and the corresponding vertex position (blue cross) for a single frame of the animation. The lines between the red and blue crosses show the correspondence between marker and vertex and the distance between them shows the error.

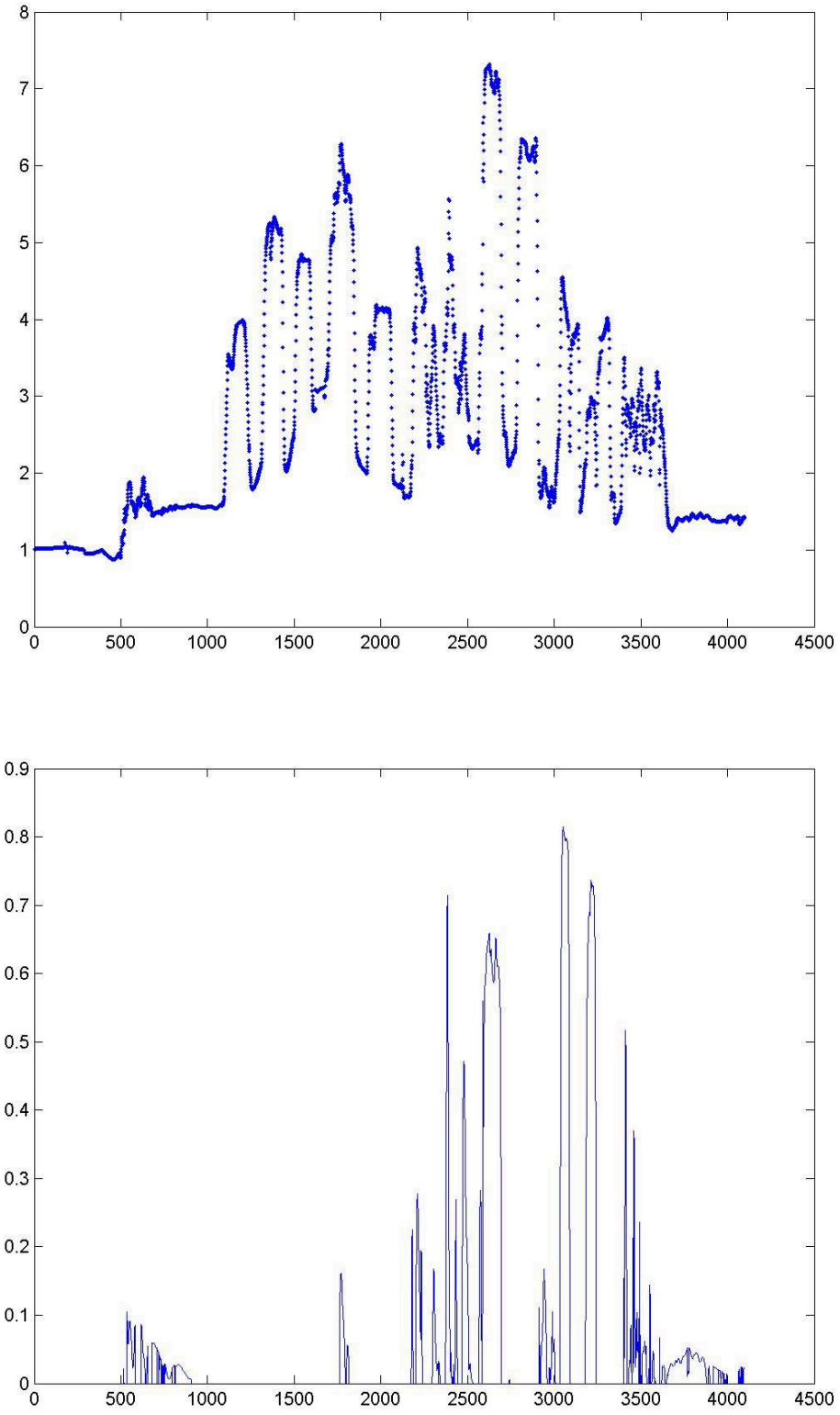


Figure 60: (Top) shows the plot of the curve for the sum of all Blendshape weights per frame of the animation. (Bottom) shows an example curve for a single Blendshape over the course of the animation. Note that both curves are continuous and smooth. Each Blendshape should ideally have a smooth weight curve over the course of the animation. This is essential as it indicates that the resulting animation is smooth and free of jitter. As a consequence of this, the curve for the sum of the Blendshape weights should also be a smooth curve.

4.1.6 Movement of the HMU and handling of spatial offsets

One challenging issue when using a head mounted unit for face capture is that there is inevitably some spatial movement of the helmet with regards to the face. This can be because the scalp moves with relation to the head, or because there is jerky movement and inertia causes the helmet to move in relation to the head rather than exactly with it. This is a well-known issue in performance capture. This can drastically affect the result of the solve as if the spatial offset isnt taken care of, it can influence the position of markers with respect to their targets and this completely throw the solve because the markers with the artificially induced distance will dominate the solve. In order to handle the spatial offsets (translations and rotations) of the helmet with regards to the face, we propose the use of dummy Blendshapes which we add to the solver to absorb the offsets or errors caused due to this.

In the case of translations, we add 6 new Blendshapes to the solve that are obtained as an offset added to the neutral shape B_0 of the Blendshape:

$$B_0 \pm (10,0,0), \quad B_0 \pm (0,10,0), \quad B_0 \pm (0,0,10) \tag{34}$$

i.e. we translate the neutral face by 10 units along the positive and negative direction along each axis. With these Blendshapes added to the mix, the solver is able to absorb the translations caused by the movement of the helmet with respect to the face and thus concentrate only on the changes caused due to the marker offsets. Figure 61 shows the effect of adding these dummy translation vectors to the Blendshape set.

In order to similarly handle the rotation caused by the movement of the helmet with respect to the face, we do the following: Let us consider the rotation about the y-axis.

If we make a shape R_1 which is a 90-degree rotated around the y-axis version of the neutral face (around some origin, say). Then

$$R_1 = \begin{bmatrix} 0\ 0\ 1 \\ 0\ 1\ 0 \\ \text{-}1\ 0\ 0 \end{bmatrix} \times B_0 + X_{Trans} + Z_{Trans} \tag{35}$$

where B_0 is the neutral face Blendshape vector. X_{Trans}, Z_{Trans} are translations on the x and z axes induced because of the rotation being about an arbitrary origin. We do not need to worry about these as the previously mentioned dummy Blendshapes for translations will take care of these.

So then we can form the Blendshape equation

$$S = B_0 + (R_1 - B_0)t \tag{36}$$

where t is the weight allocated to the Blendshape in order to account for rotation about the y-axis. We can see that $t = sin(\theta)$. For small values of θ, we can assume $sin(\theta) \approx \theta$.

Substituting equation (35) in (36), we get

$$S = B_0 + t \cdot \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \cdot B_0 + X_{Trans} + Z_{Trans} \tag{37}$$

So if we introduce that shape $t \cdot \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \cdot B_0$ as a Blendshape delta, then provided we soak up the non-nodal part by allowing for large translations using equation (34), we can have our linear solver solve for small rotations of the head with respect to the Head Mounted System in a linear fashion.

We add one shape each for all 3 axes and for positive and negative rotations. Note that for large rotations, in real life, the movement is non-linear, i.e. along an arc, so the solver cant handle it but for small rotations, the effect is negligible. Thus with a combination of the dummy rotations and dummy translations, we are able to handle spatial offsets cause by the sliding of the helmet with respect to the face.

The effect of adding dummy rotation Blendshapes on our solver can be seen in Figure 62.

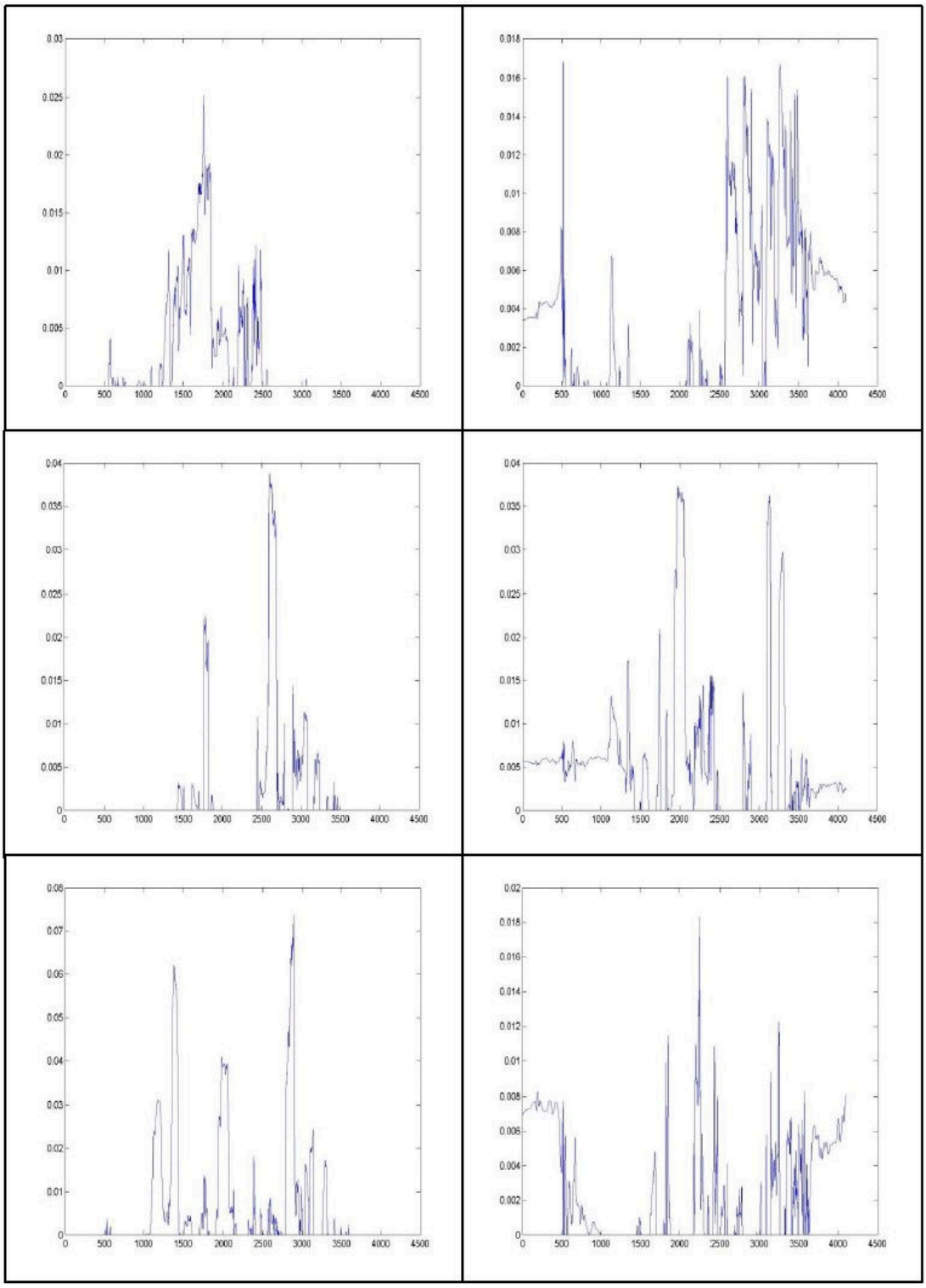


Figure 61: Graphs showing the weights assigned to the dummy translation shapes for each frame of the sequence. The weights go from 0 to 1. The 6 graphs correspond to the dummy translation shapes along the X, Y and Z axes by 10 mm on the positive and negative directions. As seen, the solver assigns weights to these shapes thus indicating that there is a need to address spatial translation caused due to the helmet moving with respect to head.

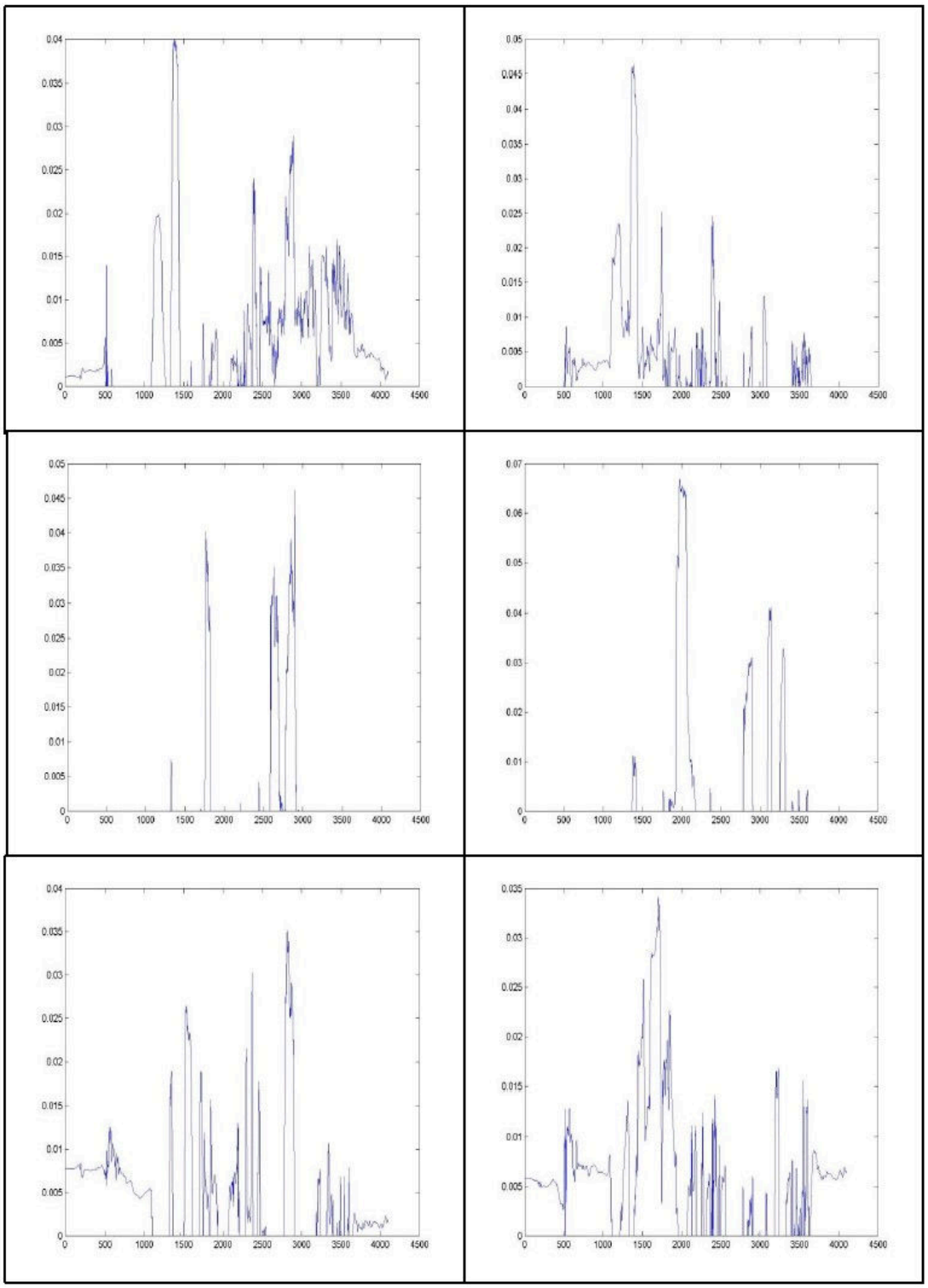


Figure 62: Graphs showing the weights assigned to the dummy shapes for each frame of the sequence. The weights go from 0 to 1. The 6 graphs correspond to the dummy rotation shapes about the X, Y and Z axes by 90 degrees about the positive and negative directions. As seen, the weights go up to 0.04, which is about 4 degrees rotation. Note: The solver can only solve for small rotations as larger rotations involve non-linear motion and will not look correct.

4.1.7 Incorporating Appearance In the Solver

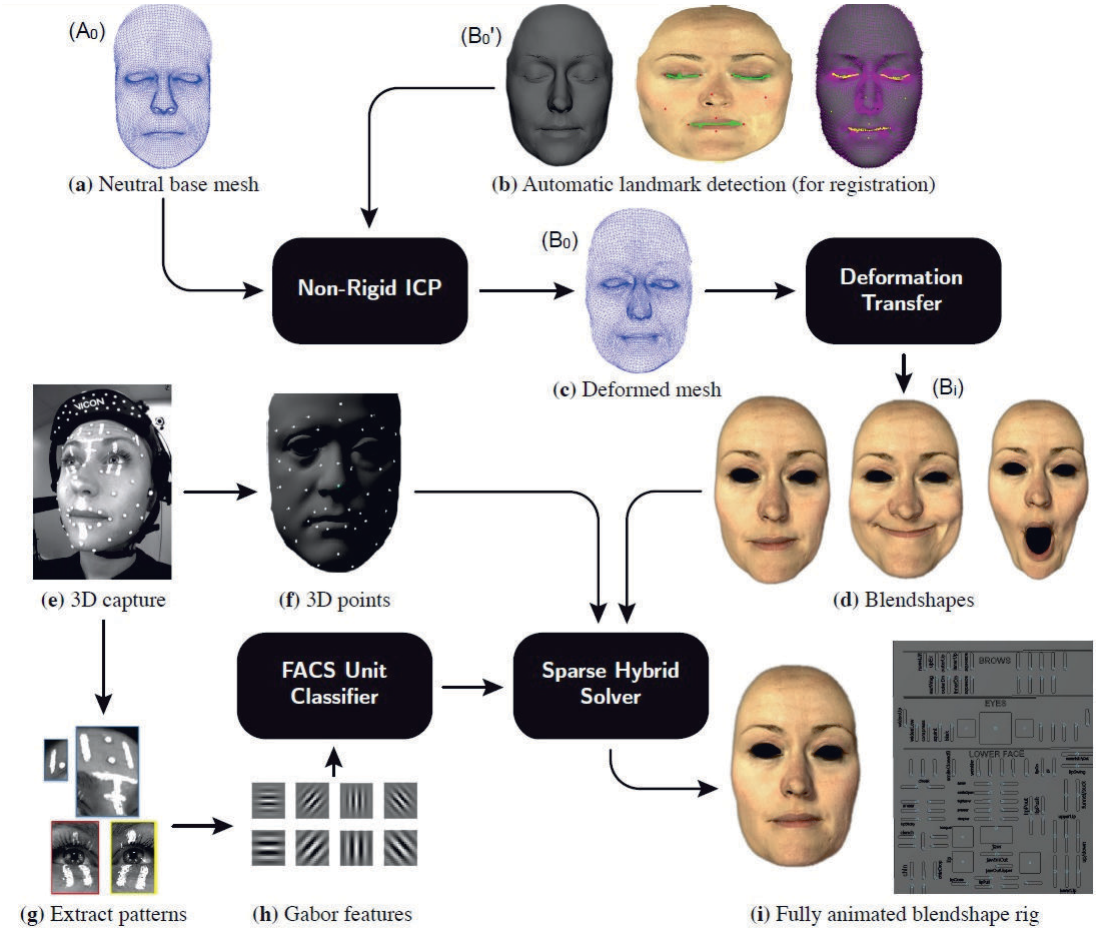


Figure 63: Overview of our modified pipeline with FACS Classification: **(a)** Given a template Blendshape model's neutral mesh we non-rigidly deform it to **(b)** a new 3D scan of a face, using an automatic landmark detection algorithm to assist with the non-rigid deformation, in order to obtain **(c)** the deformed mesh of the new face. **(d)** We then create a personalized Blendshape model using deformation transfer. **(e)** Given a new HMC performance, **(f)** 3D motion capture data and **(g)** video are acquired and used within a hybrid optimization. FACS unit classification based on the **(h)** extracted features is combined with 3D marker data to predict **(i)** optimal Blendshape weight combinations to produce a high quality Blendshape animation.

Figure 63 shows the overview of our modified pipeline which makes use of appearance information in order to improve upon the traditional marker-based approach discussed in the previous section. In our work, we extend the traditional marker-based approach to incorporate additional FACS classification from sparse make-up patterns in the video between the markers and show that this improves results.

We use the Vicon Cara system to acquire 50+ high accuracy 3D marker positions from a facial

performance. The system also provides 4 synchronized video feeds of the face. In addition, we paint extra patterns between the markers using off-the-shelf white reflective paint. This is a pragmatic decision: while facial expression recognition based on classification is a mature field, it is still not without error and unreliability on occasion due to differences in skin texture and appearance. Also areas such as the sides of the forehead and cheeks don't have much texture variation and classification in these areas is difficult. Using additional patterns greatly improves the robustness of video based classification in these regions. Figure 64 shows our head mounted system, as well as marker locations and painted patterns.

Figure 64: Video feeds acquired from the Vicon Cara HMC system.

We propose a novel hybrid Blendshape optimization (*solve*) which combines two modalities of data: traditional 3D marker data and local facial expression classification based on FACS [64] from video by utilizing the deformation of the sparse make-up patterns between the markers. Both sets of information are integrated directly into our optimizer. This allows for improved flexibility by letting just the markers drive the animation when needed and have the classification influence the result when required, thus resulting in smooth and high quality Blendshape animations. We use the term hybrid to reflect this combination of modalities. Our classifier is automated and we are able to detect different intensities of AUs. The classifier can be trained once and used on multiple performers.

Traditional solving of Blendshape weights using motion capture markers alone, does not capture the performance with complete fidelity owing to errors in the motion capture process. Production studios use 3D animators to manually add in these missing details [100]. We attempt to automate this process by looking at sparse make-up data from video between the markers and predict Blendshapes to improve visual fidelity. Traditional marker based methods work under the assumption that the solve that minimizes the objective function is essentially

the best solve. But these methods fail to take into account subtle visual cues from the video which are obvious to 3D artists. In Section 4.1.8 we analyze the objective function and discuss the factors affecting this solve.

Increasing the number of markers on the face introduces further issues in tracking. Markers can get close to each other and get mistaken for a single marker or they can be erroneously swapped causing popping in the animation. The more markers we add, the more intractable this problem becomes, necessitating manual intervention. We demonstrate that our approach can result in better Blendshape predictions, even when using a smaller number of markers. Another issue is that complex areas like the eyes and lips have frequent occlusions of markers owing to complex folding and overlapping of skin and flesh, making it difficult to track accurately. As we use a texture based classifier trained for specific facial expressions our method is able to handle these situations.

4.1.8 Hybrid Objective Function

The solution so far still only considers 3D marker data. We therefore use video classification in localized facial regions to further influence the choice of selected Blendshapes in the optimization thus making use of the region in between markers to improve the visual quality. Expression classification, particularly AU detection based on video, is a widely studied area in the Computer Vision community [114, 115]. However, to our knowledge, integrating expression classification into Blendshape solving is a novel direction in the computer graphics community. We extend our objective function to

$$E = \arg\min_{\alpha} \; E_{3\mathrm{D}} + \sum_{i=1}^{N} \gamma(\widetilde{\alpha}_i)\,[\alpha_i - \widetilde{\alpha}_i]^2 \tag{38}$$

$\widetilde{\alpha}_i$ is the (smoothed) Blendshape weight curve predicted by our classifier, where $0 \leq \widetilde{\alpha}_i \leq 1$, and is further explained in Section 4.1.12. The $\gamma(\cdot)$ term weights the influence of the video classification. It is calculated offline as a function of $\widetilde{\alpha}_i$ and it varies smoothly over the sequence. The use of the $\gamma(\cdot)$ term allows us to provide a general framework by which we can have the classifier influence the result when needed by gradually increasing the value of $\gamma(\cdot)$ or have just the markers drive the animation by driving the value of $\gamma(\cdot)$ to zero. The maximum value of this term should be in the order of the squared error of the markers so that it sufficiently influences the solve result. In our experiments we used a maximum value of 4 for this term. This parameter is calculated as follows. For every frame of the sequence, we set $\gamma(\cdot)$ to its maximum value when the classifier detects an input for which we want it to affect the results and we set it to zero when it detects an input where we want the markers to take over.

We then apply a temporal filter over the frames using a weighted moving average filter that fits a second order polynomial. We set the smoothing window size to be 15 frames. This value was set empirically. The net effect is that the solver's error will be guided by this additional term, and will therefore modify the weight of the corresponding Blendshape α_i to compensate.

4.1.9 Reflective Patterns and FACS Action Units

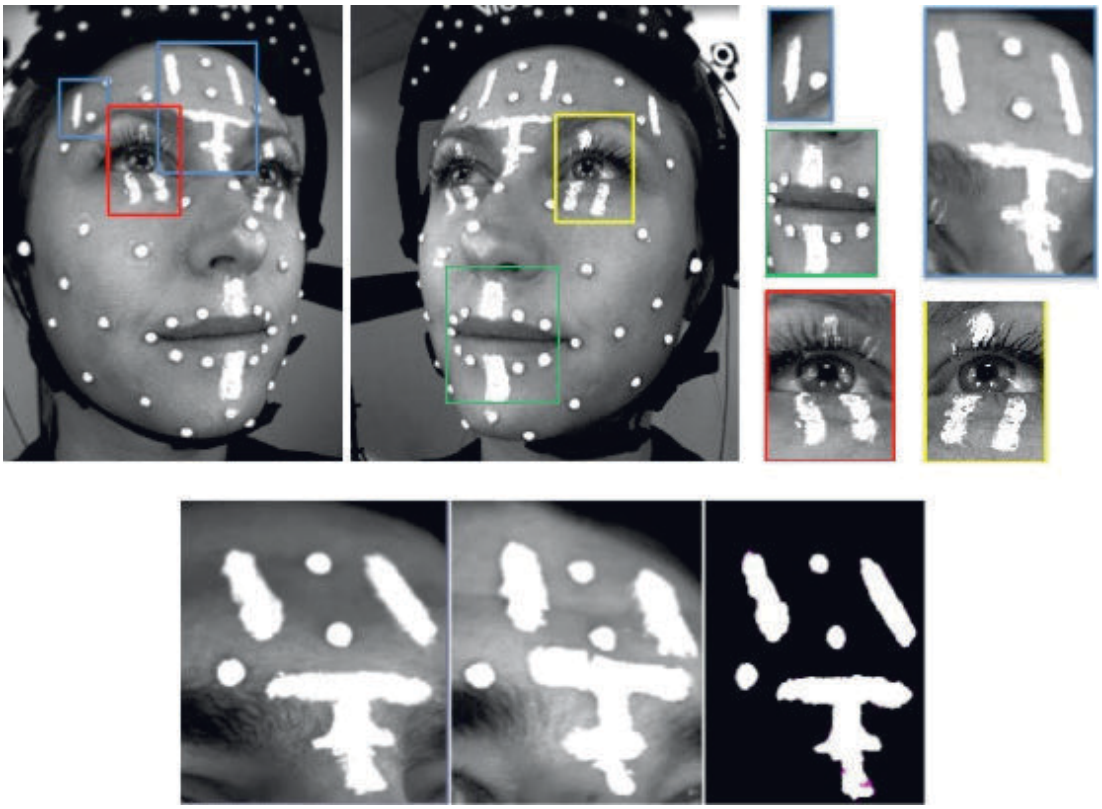


Figure 65: Video classification is performed using SVM classifiers. These are trained over a set of regions, highlighted blue, red, yellow and green in the Figure. The forehead classifier uses two regions initially, and merges their Gabor features for training and classification. The bottom row shows one of the patterns on the training subject(left), the pattern on the test subject(center) and the binary thresholded pattern after optical flow from test to training image.

As seen in Figure 65, we draw patterns on the performer's face in addition to the markers, using off-the-shelf reflective white paint. This allows us to apply a brightness threshold and only consider the pattern itself and ignore the skin texture if desired. This enables us to provide a certain level of indifference to performer identity during classification and allows us to use the classifier on multiple people. This can also be done using color paint but as our video is gray scale, we use reflective paint. Our experiments give us good classification results on different

performers but arguably training the classifier separately for each actor will give better results as the features are more specific to him/her at the cost of increased training time.

Our choice of patterns was based on muscle movement for FACS [64]. We draw patterns that capture the deformation around the inner eyebrows (AU 1), outer eyebrows (AU 2), between the eyebrows (AU 9 and AU 4). In addition, we draw patterns over the upper and lower eyelids in order to track the lid movements (AU 45 and AU 7) and handle the case of (AU 6+43), which corresponds visually to closing of the eyelids and compressing the regions around the eye. Finally we also draw patterns around the upper and lower lip to assist with lip animation.

4.1.10 Gabor Filters and SVM

In order to classify the AUs, we need to extract the relevant regions of the face and extract useful features from it. We tested our classifier using multiple features – HOG, Gabor filters and edge-detectors[116]. In our experiments, we found Gabor filters to give best classification results. We use 8 Gabor filters, at 2 scales and 4 orientations in 45° increments.

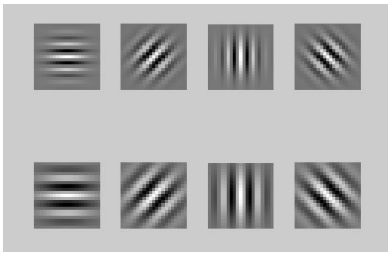


Figure 66: Gabor Filters with 2 scales and 4 orientations at 45° increments.

One important point to note is that in order for this classification to be robust, we need these regions on the face to be extracted with consistency. That means we need to track and stabilize these regions with respect to the camera. In our case, we make use of the fact that the HMC is relatively stable with respect to the head. We pick a point on the HMC that is visible in our video, and track this point through the sequence using optical flow [117] and use it for stabilization in combination with 2 stabilizing markers on the sides of the face. We found in our experiments that this results in good stabilization of the face with respect to the camera and lets us extract these regions accurately.

4.1.11 Training the Classifier

For the training phase, we ask the performer to perform 7 AUs around 4-5 times each. We then extract the regions of the face from the video. Figure 65 highlights facial areas of interest, as

well as the distinct painted patterns. The images are thresholded to extract the reflective patterns. For each AU and each region, we separately perform K-means-clustering on the largest mode of variation in the video-texture, resulting in 4 clusters which correspond to 4 intensities of activation. These clusters are used to label the data for training. Figure 67 shows the intensity levels obtained using this approach for AU 1+2. We then apply Gabor filters on these extracted images to get our feature vectors and perform a Principal Component Analysis (PCA) for dimensionality reduction. The PCA retains the basis that capture 90% of variance in the features. Finally we normalize our training data and train the SVM using a linear kernel. Our features are large in dimension (2x4xNumOfPixels) and hence a linear-kernel gives sufficient separability as evidenced during cross-validation with accuracies of 98%. Using an RBF-kernel didn't improve performance. We use the one-vs-one approach for multi-class classification.

Thus, each SVM is trained on labeled data for each action in that region. Our classifier was trained to detect AU 1+2, AU 7, AU 4, AU 9, AU 45 and AU 6+43 on the upper face. Also, in order to demonstrate the applicability of our method for improving lip animation, we trained our classifier to detect the lip pucker combined with a sideways motion (AU 10(L/R) + 12(L/R) + 18(L/R) + 23(L/R)) [64], as an example see Figure 69 (right).

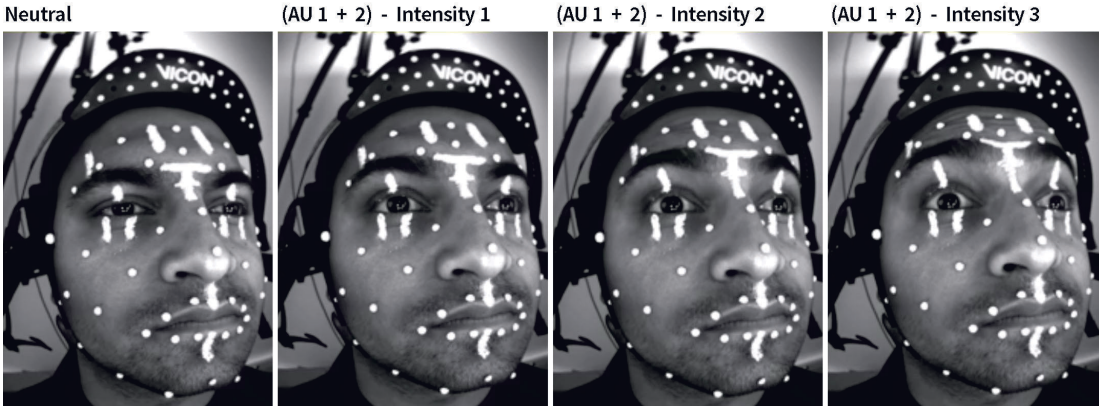


Figure 67: K-means clustering is performed on individual Action Units which clusters the motion into 4 groups corresponding to 4 intensities. These labels are then used to train the classifier. The image shows 4 intensities for AU 1 + 2 automatically obtained using this approach.

4.1.12 Classifying Action Units

Given a performance, we extract the regions from the face and process them in the same manner as during the training phase. In order to account for possible differences in the extracted regions between the training and testing videos and for slight variations in the patterns, we first make

sure to resize the extracted test images to be consistent with the training data and then also perform dense non-rigid image alignment to the neutral pose of the training subject using optical flow [118]. This gives us a UV flow field for the neutral pose, which is applied to every frame of the performance in order to adjust for the variations in the pattern shapes. This is shown in Figure 65 on the bottom row.

The output of the SVM classifier predicts which FACS action units are triggered and at what intensity, for every frame of the performance. The mapping between the FACS action units and the Blendshapes is trivial and needs to be done only once per rig. The Blendshape weights predicted by the SVM are still discrete. In order to make these continuous, we apply a smoothing function. We use the Savitzky-Golay filter in Matlab which is a weighted moving average filter that fits a polynomial of a specified order over a specified number of samples in a least-squares sense. We found this to be better than using a simple averaging window as it preserves high frequency data better. We used different smoothing-window sizes for different AUs, ranging from 20-30 frames and a second order polynomial. These were chosen empirically, and consistent across subjects. We then normalize these values between 0 and 1, thus getting continuous weight values $\widetilde{\alpha}_i$ over the sequence for the Blendshapes. We use these blendweight predictions $\widetilde{\alpha}_i$ as mentioned in equation 38.

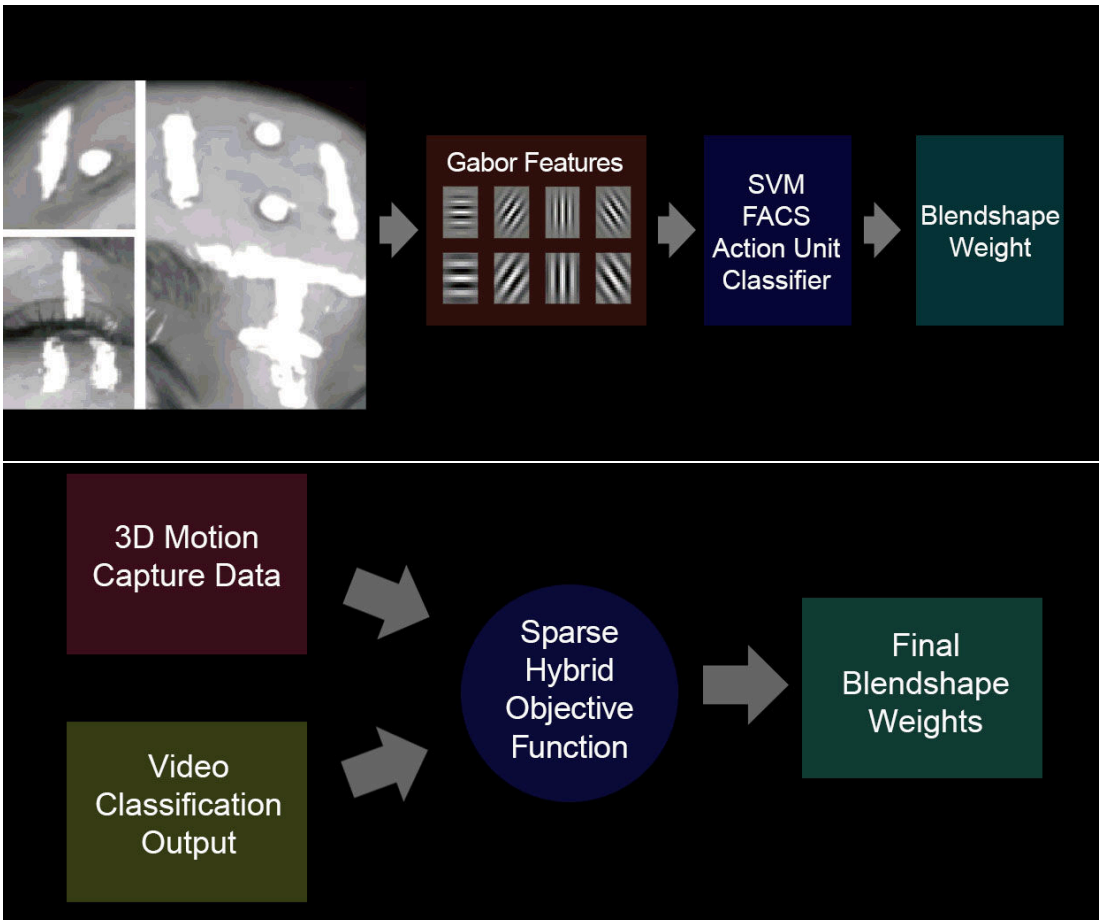


Figure 68: Training the classifier using Gabor features as input to the Support vector machine. The classifier outputs Blendshape weights that correspond to FACS units detected in the image. The results of the classifier are combined with the motion capture input in the hybrid objective function to give us the final Blendshape weights for the frame.

4.1.13 Results - Using Markers and FACS Classification

In this Section we compare example frames from animations solved using our method with those from methods using purely marker based approaches. The results for our purely marker based outputs were generated using equation 33, which is standard. The accompanying supplementary video material gives an overview of our system, and shows multiple animation results with comparisons.

Hybrid Solver: We used one of our participants to train our bank of SVM classifiers, as described in Section 4.1.11. We then captured the same performer and two others carrying out a range of facial expressions and dialogues. Figure 70 shows example video inputs from the HMC and corresponding frames from the resulting animations, using our method and for purely marker based approaches. In the bottom row, notice how the addition of the FACS classification affects the regions between the eyebrows (AU 4 and AU 9). These differences are very subtle but completely change the way the expression is perceived. These subtle differences are not captured using markers alone. Although the normal maps obtained from the high resolution scans are baked into the corresponding Blendshapes and trigger when the corresponding Blendshapes are activated, the markers by themselves don't drive the Blendshapes accurately resulting in subdued expressions. This is caused by a few factors. As mentioned in Section 4.1.2, we find the optimal barycentric co-ordinates for the markers based on an iterative error minimization over a ROM sequence. In spite of this, the markers may not attach themselves to the exact location on the mesh corresponding to their location on the face during the rigid-alignment phase. This problem is especially exacerbated when the mesh is low resolution. One solution is to manually modify the position of the marker on the mesh by visually inspecting its position on the face. This is reasonable in locations that are visually discernible like the tip of the nose and lip corners but difficult in areas without distinguishing features like the forehead and cheeks. Also this gets prohibitive as the number of markers increases. Another factor is that because the individual Blendshapes are generated from the template model using deformation transfer, there is an inherent scale error in that the range of movements of the subject do not match precisely with that of the model. On the other hand our method makes use of the additional texture information and the classifier is able to detect the deformation in the patterns accurately. It is able to recognize the FACS units and gauge their intensities exactly and influence the Blendshape weights such that the expression is recreated correctly and the normal map blended in appropriately.

Figure 69 (right) shows an example of our method being used to improve lip animation. The markers alone aren't able to capture any information about the inner lips and are oblivious to the fact that the lips are closed and hence the solver gives an incorrect result. Our method on

the other hand is aware of this pattern deformation as it has been trained to detect it and hence predicts correct weights. The accompanying video shows the same animations, which show themselves to be both high quality and visually close to the input videos in terms of expression and speech motions.

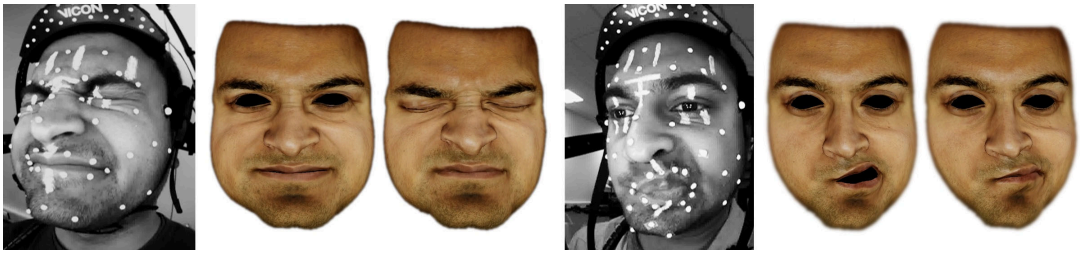


Figure 69: Our system solves for the optimal Blendshape combination from motion-capture data by considering both 3D markers and video based FACS classification from sparse make-up patches between the markers. The middle image for each 3-tuple shows solutions to the Blendshape model without video classification, while the right image for each tuple shows solutions using video based FACS classification to guide the optimization. Our method is able to capture information from the video that markers alone aren't able to capture accurately, such as the region around the eyes in the left tuple and the lips in the right.

Adding more markers: In order to assess whether our result using 3D markers alone wasn't optimal due to there not being enough present on the performer, we conducted a second experiment. We applied 54 markers to the upper face of a performer alone, and few more on the lower face. Figure 71 shows still images of the performer and corresponding Blendshape model output, while the accompanying video shows an animation of the corresponding sequence. It is clear that even with a dense set of markers on the face, the 3D only solver does not capture all the detail. Subtle motions like the furrow between the brows (AU 4) and challenging expressions like AU 6+43, are not captured using markers alone, while our method is able to capture these.

4.1.14 Discussion

In our experiments, we trained our classifier to detect only a few AUs. Of course this can be extended to as many isolated AUs as the performer can train for. Adding more AUs to our system implies that we have to consider combinations of these AUs during training. Note that while we'll need to train for these combinations we can choose to have the classifier output affect the solve just for a few desired combinations and have only the markers handle the rest. Given good training data that covers the general variations within a particular movement, the classifier is able to reliably handle these when solving for the performance.

Figure 70: Animation Results: The left-most image of each 3-tuple shows an example image from our HMC. Middle images show results using only markers. The right images show results using markers combined with video classification. Our method is able to capture subtle motions between the eyes such as AU 9 (top-left tuple) and AU 4 (bottom row) which are missed when using only markers. This drastically changes the way the facial expession is perceived. Also increased control over Blendshape selection allows us to detect when wrinkles should show up (top-right and center-left tuples)

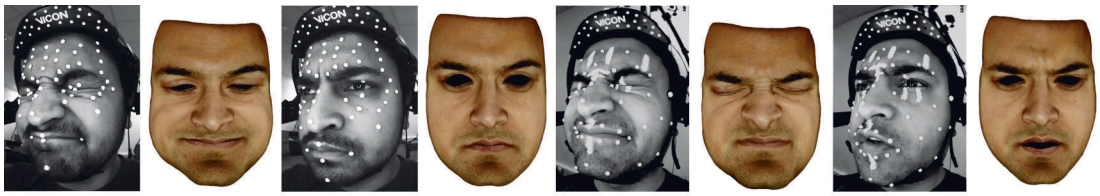


Figure 71: Example comparison using significantly larger number of facial markers (54 in the upper face region alone). Left 2 tuples show the marker-only approach while the right 2 tuples show our approach. As seen, even when using significantly larger number of markers in the concerned region, it still results in missing facial detail when using only markers which is otherwise captured when using our method. Results are shown for AU 6+43 and for AU 4.

The strength of our approach lies in the fact that we can have just the markers drive the animation in general but also have the classifier influence the result for more challenging motions. In

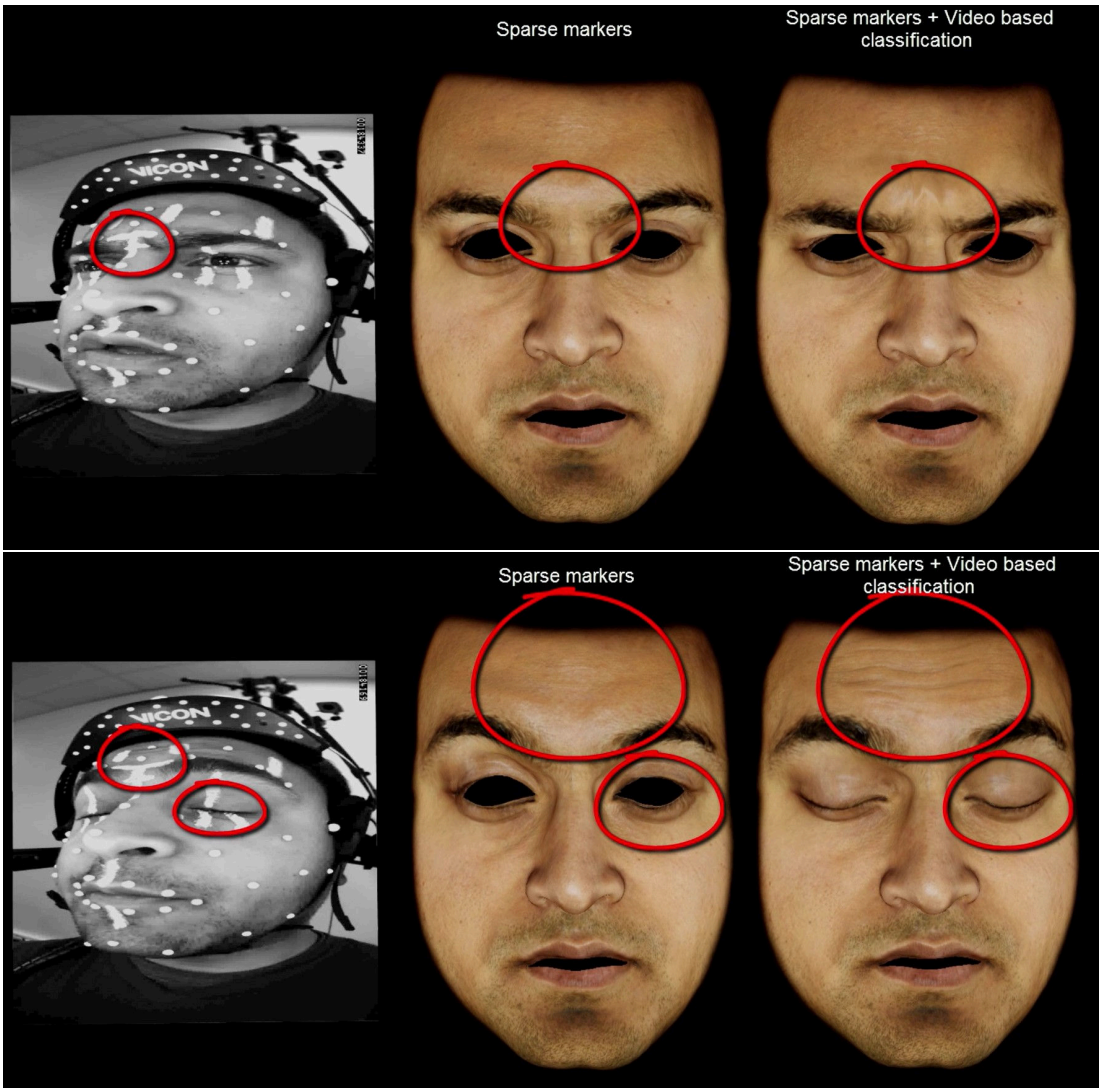


Figure 72: Result comparing the solve using just a traditional marker based solve vs our method.

order to do this we make 2 passes. In the first pass, we use the classifier to predict Blendshape weights from video as described in Section 4.1.12 to obtain weight curves $\widetilde{\alpha}_i$. The weighting factor $\gamma(\cdot)$ is calculated as described in Section 4.1.8. In the second pass, we use the curves from the first pass and solve equation (38) to obtain the final weights. We use the quadprog optimizer in Matlab with the interior-point-convex algorithm to minimize our objective function and impose the linear inequality constraints on α_i. As we smooth over the classifier weight outputs, our method is not real-time.

For our purposes, we want to detect the presence or absence of certain poses and have the

Figure 73: Figure showing the retargeting of our blendhape weights onto multiple Blendshape rigs using parallel parameterization.

classifier affect the results when needed. The use of a classifier lets us provide a general framework to allow this when used in conjunction with the $\gamma(\cdot)$ parameter. In theory a regression based approach can be used to achieve the same effect but we'll nevertheless have to make a few choices about the complexity of the model and the values of thresholds which amounts to the same choice as the smoothing window size for our $\gamma(\cdot)$ and $\widetilde{\alpha}_i$ parameters when using classifiers. Noise in the input data would be another factor to consider which may necessitate a smoothing operation on the predicted output curves just like in the classification case.

As discussed in section 4.1.9, we extract the patterns in the texture thus enabling the classifier to work on multiple people independent of identity. As expected, the accuracy of the classification degrades slightly when we train on one participant and use it on another, in spite of the non-rigid image alignment between subjects. This is due to sensitivity of the classifier to local transformations occurring due to inherent differences in the motion between participants. This is especially noticeable around the eye region as it has the most variation between subjects. This manifests itself as misclassified frames causing inconsistency between actual performance and recreated animation. This issue can be alleviated by making the classifier invariant to local

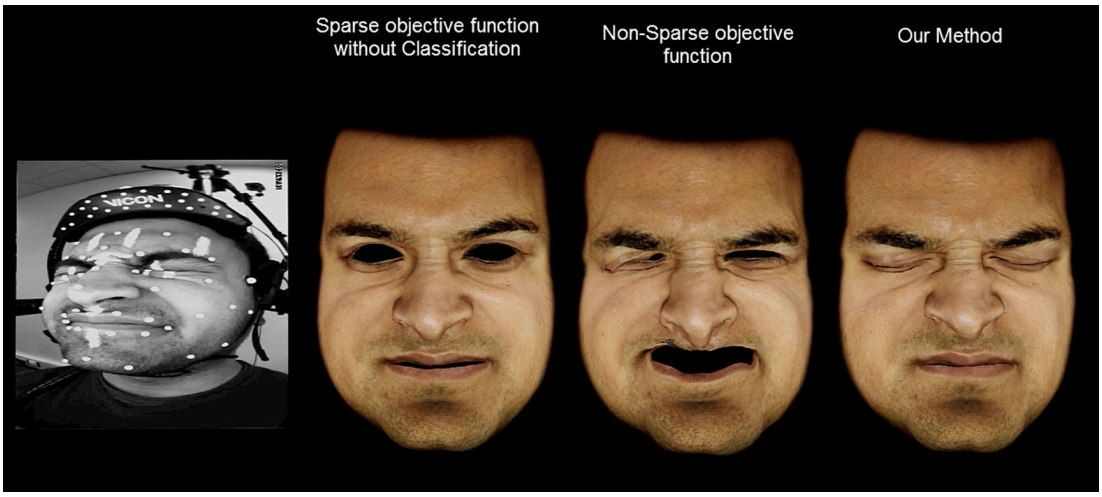


Figure 74: Results showing the effect of the sparity constraint (Section 4.1.4) on the solve. Note that the lack of the sparsity constraint causes overfitting, where the solver tries to add in more Blendshapes to account for minute errors between markers and vertex-points and adds in additional Blendshapes to account for these. This causes distortions as seen in the result (Middle).

transformations of the input. This can be done by augmenting our training data with random locally transformed replicas of the training patterns at the cost of increased training time or by using more robust classifiers that have the invariance property built into them such as in convolutional neural networks. Ideally the classifier should be trained on AUs from multiple people. There are limitations to this and AUs for subjects with drastic differences in scale or whose FACS movements are very different compared to the training data can be misclassified. In this case, the classifier will work better when trained specifically on the individual.

4.1.15 Conclusions and Future Work

We have presented a novel method that uses information between markers in the form of sparse make-up patterns in the video and classifies FACS units in order to better fit Blendshape models to facial performances. Our approach guides the overall optimization function to include movements difficult to detect using 3D motion capture alone. Our resulting animations are high quality and effectively parameterize the actions of the performer. We have compared our hybrid solving approach to traditional motion capture methods that use only 3D markers and shown that our results are more faithful to the performance.

Our method can be extended to handle dimples and other micro-expressions in the future. Although we have used SVMs in our method, we plan to consider classifiers that might be

more optimal such as Relevance Vector Machines or Deep Learning architectures with the aim of improving robustness of classification.

We believe there is also room to improve our method to allow detection of AUs without special makeup. This is still however an area of research in computer vision, especially given captures in environments with broadly changing lighting variation. However, recent work in machine learning for AU detection [114] shows promise in this area, and may allow for the recognition of many subtle motions across a wide variation of performers.

4.2 Markerless Capture System

Over the last few years many methods for facial performance capture have been developed. These methods range in complexity from active marker based approaches with multiple cameras, head-mounted devices and controlled environments [24, 7, 27, 89, 28, 8, 5, 31, 119, 49, 25, 39, 56, 74] to methods that use depth sensing devices like the Microsoft Kinect and others [52, 53, 96, 30, 37]. Although many of these methods provide good quality animations, the time taken to setup the environment including the time taken to apply markers on the actor's face can be a hindrance to the quick application of these methods, similarly the capture devices used may be custom built and not easily available or cheap enough for general consumer use. The use of markers or structured light makes it difficult to simultaneously capture both geometry and texture thus requiring either in-painting of markers as in [48] or sacrificing temporal resolution by interleaving structured light with uniform light. Additionally these active methods can be uncomfortable for the actors and affects their performance [8].

As stated by Chris Bregler [23], "In a way, Makerless Face Capture for animation is as old as the trade of traditional animation, taught by Disney and others in the early 20th century. Animators always look at example recordings, especially for facial animations. For example, the idiosyncrasies and mannerisms of the Step-Mother's facial expressions in Cinderella were based on filmed recordings of actress Eleanor Audley. In some cases, the animations were even a direct copy of the hand-traced motions from film, like in scenes of Snow White. Performing such tracing and tracking automatically by a computer has been the topic of computer vision research over the past 20 years. Just recently new techniques have become available that have good enough quality for animation in the entertainment industry." Several companies have come out with systems for markerless face captures, including Mova Contour [120], Image Metrics Live Driver [99], DI4D [113].

In general, there seems to be a trade-off between the quality of capture data and the complexity of the hardware or setup process. On the other end of the spectrum are passive, single camera systems and these have been the focus of research in recent times [93, 90, 33, 34, 35, 36]. Due to the challenging nature of uncontrolled lighting environments and unreliable textures, the tracking is usually limited to distinctive facial features such as eyes, eyebrows, pupils, inner and outer contours of the lips and contours of the jaw. Most faces do not have sufficient medium-scale texture to establish dense correspondences in different viewpoints or across frames.

That being said, there is ample data that exists from legacy systems, easily available, captured from multiple people performing a range of facial movements and speech. One of the problems with 2D monocular inputs is that information along the depth axis is inherently missing and this proves to be a challenge. Even though the solver may minimize the error in the ob-

jective function, the true 3D shape may be ambiguous. In the following sections, we present a lightweight approach that achieves good quality solves using just a monocular input while exploiting this existing data from legacy systems and using it to learn a prior in order to make sure physically implausible results aren't generated. Our contributions are as below.

- We present a lightweight monocular markerless capture method that achieves good quality animation parameters and does not require special equipment, controlled lighting environments or complex training phases.
- We exploit easily available prior animation data obtained from 3D tracking systems and use this in a density estimation framework to regularize our objective function and generate more plausible results. We enable further flexibility by learning separate prior constraints for the upper and lower face regions.
- We combine initial estimates of 2D landmark points on the face based on an ensemble of regression trees, with an Active Appearance Model for improved accuracy.
- We handle noise in input 2D features and in the estimation of camera extrinsic parameters thus eliminating jitter in the resulting animation.
- We also present our lighting optimization approach that uses the inverse rendering equation to find the optimal values of light intensity and position thus enabling us to relight, re-texture and overlay the mesh on the original video.

In this work, our objective is to capture the movement of the actor's face using a lightweight method that doesn't require expensive equipment or lighting apparatus and using only a single monocular off-the-shelf camera while simultaneously addressing the issue of missing depth information. In light of this, we will mainly discuss previous work that is similar to our approach in these regards.

Our approach is most similar in spirit to Garrido et al. [36]. They present a lightweight approach that makes use of a single monocular video input and are able to generate a high quality output animation with fine scale details using a Blendshape model. They track a few sparse landmark feature points on the face reliably using forward and backward optical flow combined with automatic key-frame selection based on local binary patterns for robustness. The pose and facial expressions are estimated from these sparsely tracked points on the face in an iterative fashion. Temporally coherent dense motion fields tracked from video combined with a smoothness constraint is then used in order to refine the pose and facial expression. Fine scale details are then added on top using a shape from shading approach. Although their method produces good results, as it is a monocular system, it doesn't account for inherent loss of information along the depth dimension. Our method uses a prior constraint in order to regularize the

data which alleviates the error due to the lack of depth information. Also our method doesn't depend on optical flow across the sequence, but only on the current and previous frames and thus can be implemented in an online fashion.

4.2.1 System Overview

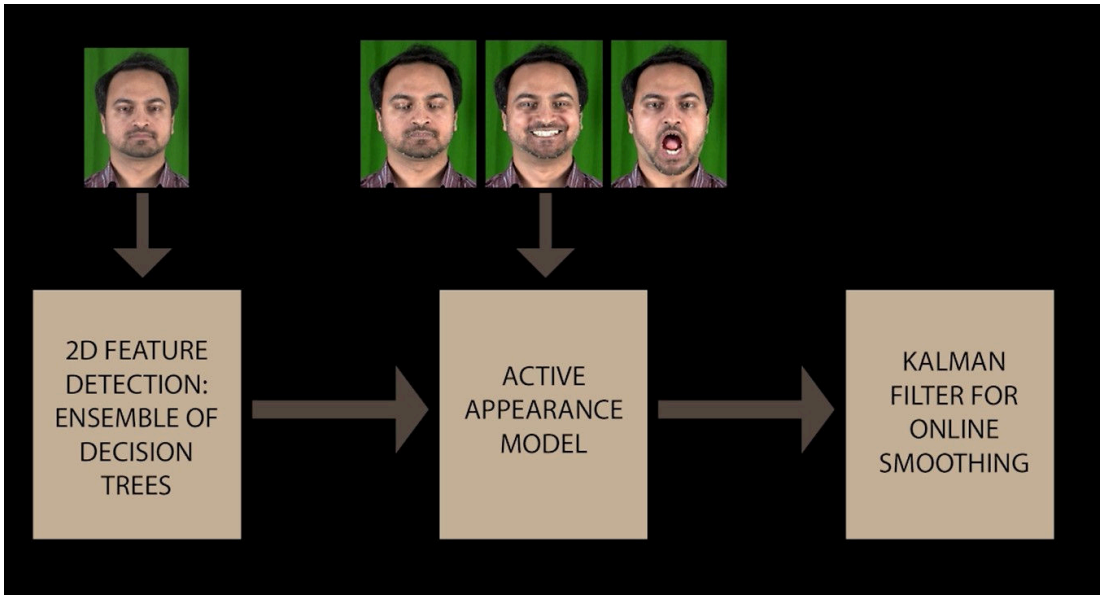


Figure 75: Our system uses the input from a 2D feature detector based on an ensemble of decision trees as an initialization and further feeds it to an Active Appearance Model trained on 15 images of the user with varied facial expressions. This allows us to accurately detect the features on the actor's face even during extreme facial expressions. We then feed the result of this step into a Kalman filter for online smoothing. We use 2 Kalman filters, one for the 2D features and one for the camera's extrinsic parameters.

Our pipeline fits a Blendshape model to automatically tracked 2D feature points in the video. In order to generate our Blendshapes (Sec.4.2.3), we first deform the neutral expression mesh from an existing template Blendshape model to a 3D scanned face of the actor to obtain the mesh of the actor in a desired topology. We then automatically generate the user-specific Blendshape expressions for this actor. Our 2D landmark features are initially obtained using the method of [103] giving us 68 distinct landmarks (Sec.4.2.2). These landmarks are detected per frame and although they provide a good starting point, the detections are not accurate enough for our purposes and give unacceptable results especially for extreme facial expressions. In order to address this, we use the Fast Simultaneous Inverse Compositional algorithm of [121], trained on a few images of the user which is applied on top, using the results from [103] as an initialization. This gives us more robust landmark detections especially in extreme facial expressions. We solve for the camera extrinsic parameters using the Perspective-n-Point approach with the Levenberg-Marquardt algorithm for non-linear optimization (Sec.4.2.4). In order to address the inherent noise from estimating landmarks and camera parameters per frame, we smooth the noise in the 2D features and the camera parameters using a Kalman filter

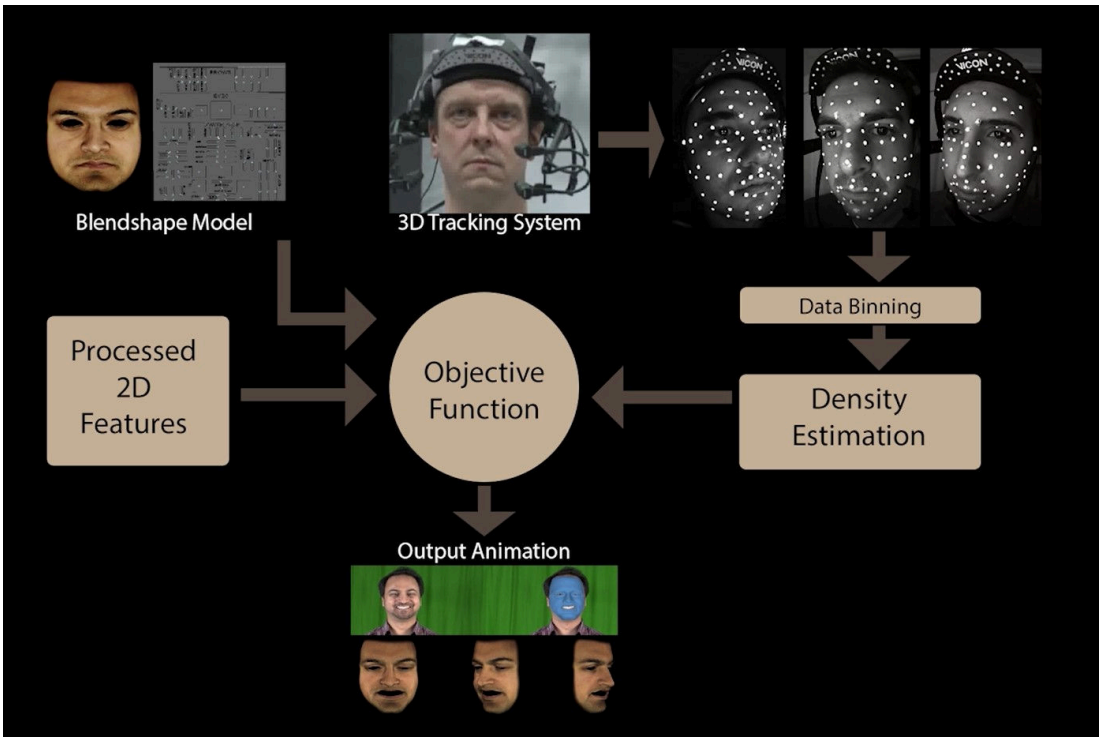


Figure 76: Our system makes use of prior data in the form of existing animation sequences obtained from previous captures using accurate 3D tracking systems. This includes multiple facial expressions and speech movements across multiple people. We learn this prior within a density estimation framework on existing data, using a Parzen window scheme with a radial basis kernel. We also perform a data binning operation in order to make sure that the density estimate isn't affected by the frequency of the data points in the prior data.

(Sec.4.2.5). Finally in order to regularize our solve results, we make use of a prior energy constraint (Sec.4.2.7 and Sec.4.2.8) that estimates the probability of the solution and factors it into the objective function (Sec.4.2.9) resulting in more plausible shapes. Our optimization function then solves for the optimal coefficients of the Blendshapes that minimize the re-projection error of the landmark vertices whilst taking into consideration its likelihood. Finally, we use an inverse rendering equation (Sec.4.2.11) to optimize for the lighting parameters that allows us to relight, re-texture and overlay the face on the video.

4.2.2 2D Facial Feature Detection

To obtain our initial 2D features, we first use the algorithm of [103] implemented within the Dlib library [122], to detect 68 landmarks on the face on a frame by frame basis. The algorithm uses an ensemble of regression trees to accurately predict the optimal displacement of each landmark at each level, based on differences in pixel intensity around that landmark. We trained the ensemble of regression trees using the images from the HELEN, LFPW, AFW, IBUG and 300 faces In-the-wild databases [123, 124, 125], to give a total of 4213 training images with a wide variety of pose, lighting and shape variations. We used a cascade depth of 10, tree depth of 10, 500 trees per cascade, a feature pool of 400 and 50 test splits (see [103] for details). This gives us reasonable initial detections of landmarks on the user's face but isn't accurate or robust enough for the application of performance capture as shown in the accompanying video. The landmark detections are incorrect during extreme expressions and this leads to unacceptable solve results.

In order to improve upon this, we further train an Active Appearance Model [29] using the Fast Simultaneous Inverse Compositional approach of [121], on a few select images of the user performing a few facial expressions. We used 15 facial expressions that included a few that elicit the extreme range of the user's facial movements (jaw-open, lip-swing) and also a few challenging expressions that generate lip and eye occlusions (pucker, squinch). We then use the result of the previous step as a starting point and then use the AAM to improve landmark tracking through the sequence. In our experiments, this gives us much more robust detections and also covers the full range of the user's expressions as shown in the accompanying video. The results of this landmark detection are further fed into the Kalman filter to account for discrepancies between frames and to remove noise in an online fashion.

Figure 77: Landmark points initialized using an ensemble of regression trees and updated by the AAM

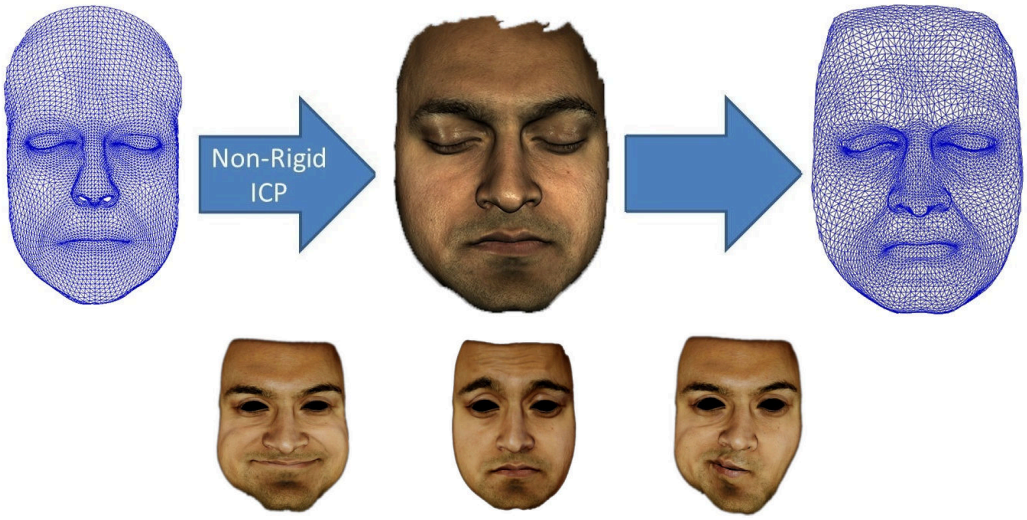


Figure 78: Non-rigid ICP from template mesh to scan resulting in user specific Blendshapes

4.2.3 Automatic Blendshape Generation

The Blendshape generation step for obtaining the actor's Blendshapes is exactly as described in Section 3. We briefly describe this process here again. In order to obtain the neutral expression mesh of the actor, we scan the actor in a one-time pre-processing step. The resulting scanned mesh has a random noisy topology which cannot be used directly. In order to rectify this, we then deform the neutral expression mesh from an existing template Blendshape model to this scanned mesh using the algorithm of [11] to obtain a new mesh with the desired topology. We then automatically generate the expression Blendshapes for the actor's face by applying the Deformation Transfer approach of [16] to give us 140 unique Blendshapes. This gives us a linear Blendshape model with N Blendshapes for use in our solver as below:

$$B = B_0 + \sum_{i=1}^{N} \alpha_i (B_i - B_0) \tag{39}$$

where, B_0 is the neutral expression Blendshape, α_i is the weight associated with Blendshape i and B_i corresponds to the i-th Blendshape.

The vertices on the 3D mesh that correspond to the automatically detected landmarks from Sec.4.2.2, are chosen by the user in a one time manual step.

4.2.4 Camera Calibration for Facial Projection

We calibrate our camera using a standard checkerboard pattern in order to obtain our camera intrinsic matrix K in a one time step. We then calculate the extrinsic parameters $[R|t]$ i.e. the rotation and translation that take the 3D face mesh from model coordinates to camera coordinates. This is essentially a Perspective-n-Point problem between the vertex coordinates on the mesh and the corresponding landmarks in 2D. The calculation of the projection matrix has to be independent of the facial expression of the actor and thus we choose a subset of the landmarks that correspond to stable regions on the face. This includes the landmarks on the upper/outer contours of the face and along the bridge of the nose, as shown in Figure77. In our experiments this gives us satisfactory results. Given these correspondences, we then solve for the camera extrinsic parameters every frame using the Levenberg-Marquadt non-linear optimization framework combined with RANSAC for robustness. This gives us the $[R|t]$ values every frame that we combine with the camera intrinsic matrix K to give us our projection matrix – $K[R|t]$. In order to account for differences between adjacent frames, we further process these results and smooth them across frames by feeding these into the Kalman filter.

4.2.5 Online Facial Feature Smoothing

The 2D features and the camera extrinsic parameters so far, are obtained on a frame by frame basis. Both the 2D landmark detections and the camera extrinsic estimates are prone to noise and this independence between the frames inevitably leads to jitter in the final animation as shown in the accompanying video. This necessitates a smoothing operation in order to ensure consistency and avoid sudden changes. We use a Kalman filter in order to smoothly transition these values between frames. The Kalman filter predicts the value of the landmarks and extrinsic parameters every frame and then uses the observations to update its belief about what the parameters should actually be, based on the value of the Kalman-gain-factor, which it calculates based on the noise in observation and the noise in the process. This ensures that any updates made to the landmarks and the extrinsic parameters are updated smoothly. We use one Kalman filter for updating the changes in all the 2D landmarks and one for updating the changes in rotation and translation parameters for the camera extrinsics.

The location of the landmark point detections are predicted every frame by taking into account the velocity and acceleration along each dimension of the tracked points based on the previous frames and then combined with the observations at that frame. A similar process is applied for the translation values in the camera extrinsics. As for the rotations, the orientations in 3D

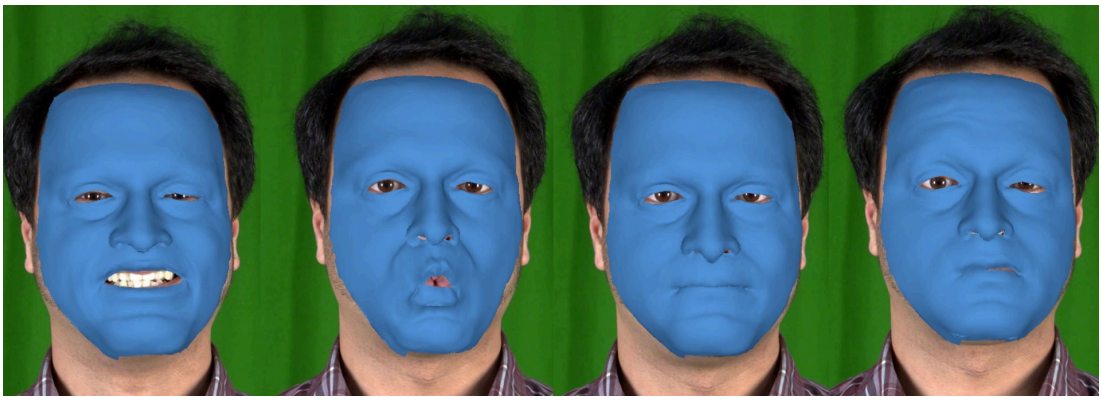


Figure 79: Results from the projection of our 3D face mesh onto the video

have 3 degrees of freedom (yaw, pitch and roll) but our rotation matrix has 9 parameters. Smoothing the rotation matrix directly in this space will not constrain the values properly and will not ensure valid rotation values. Hence we perform the smoothing in quaternion space. This removes ambiguity and also constrains the rotations better thus ensuring that our rotation values are valid throughout the sequence.

This process of online smoothing using Kalman filters greatly helps with tackling noise inherent in the 2D feature detection and camera extrinsics and ensures that our objective function isn't affected by noise. The results of our updated landmarks and camera projection matrix can be seen in Figure 79 and in the accompanying video.

4.2.6 2D Objective Function

Given the corrected 2D landmarks and the projection parameters from the previous step, we then solve for the facial expressions by calculating the optimal coefficient weights for the Blendshape model using the following objective function, similar to [33].

$$E_{2D(\alpha)} = \sum_{l=1}^{n} \|\Pi_Q(M(B_0 + \sum_{i=1}^{N} \alpha_i B_i)^{(v_l)}) - q^{(l)}\|^2 \tag{40}$$

where:

- n is the number of landmarks
- Π_Q is the camera projection matrix
- M is the rigid transform from object space to camera coordinates

- N is the number of Blendshapes
- B_0 is the neutral expression Blendshape
- α_i is the weight associated with Blendshape i with the constraint $0 \leq \alpha_i \leq 1$
- B_i corresponds to the i-th Blendshape
- $q^{(l)}$ represents the l-th 2D landmark point in the image
- v_l represents the vertex corresponding to landmark l

This objective function by itself only considers the 2D landmarks to reduce the error and is inherently under constrained as the information along the depth axis is missing in the video. This leads to insufficient constraining and to errors that show up especially along the depth axis. In Section 4.2.7 and 4.2.8 we tackle this problem by improving on our objective function and making use of prior constraints.

4.2.7 3D Spatial Constraints

One of the problems inherent in monocular capture approaches is that the information along the depth dimension is lost. This leads to situations where the optimization function described above is able to obtain coefficients that minimize the squared distance between the projected vertices and the corresponding 2D landmarks but it does so with solutions that no longer adhere to physically plausible shapes of the face. This effect is especially visible around the mouth region as there is a lot of variation in the depth dimension during speech. This can also be seen when opening the jaw as there is movement along the depth axis. In order to ensure that our objective function provides physically plausible results, we need to ensure that our results are regularized to stay within such a solution space. In general 3D face capture systems can be less flexible compared to monocular marker-less systems and placing physical markers on the actor can be very time-consuming, but these systems provide accurate tracking of points. There is ample data available from legacy 3D face capture systems from multiple people performing different facial expressions and speech sequences. It makes sense to use this data in order to regularize our results. We can use the data from these accurate 3D systems and use it to constrain our results while still retaining a monocular marker-less approach.

In a one time training step, we use prior data in the form of 3D marker locations over capture sequences from previous captures using the Vicon Cara 3D head-mounted device [7]. Our data spanned multiple sequences of speech and facial expressions from 5 different people performing diverse facial movements. In order to use this data for our purposes, we first need to solve

for the Blendshape coefficient weights specific to our Blendshape model. We do this using the standard objective function [17] for 3D solves, as shown below.

$$E_{3D(\alpha)} = \arg\min_{\alpha} \|B_0 + B\alpha - T\|_2^2 + \beta\|\alpha\|_1 + \alpha^T \Gamma \alpha \tag{41}$$

where:

- B_0 is a $3n \times 1$ vector representing the neutral face, where n is the number of markers.
- B is a matrix of size $3n \times N$, that contains the deltas for each of the Blendshapes $B_{i...N}$, where N is the number of Blendshapes.
- α is a $N \times 1$ vector of weights with the constraint $0 \leq \alpha_i \leq 1$.
- T is a $3n \times 1$ vector representing the target markers.
- The term $\|\alpha\|_1$ is an L1-norm on α that penalizes the sum of weights.
- β is a weighting factor on the L1 regularizer.
- The Γ term is a Tikhonov regularizer that ensures that the function is convex and has a unique global solution. $\Gamma = \varepsilon I$ where ε is a very small constant and I is the identity matrix of size $N \times N$.

Solving this for our multiple training sequences gives us valid coefficient weights over multiple people. We use this data D_{prior}, in order to estimate the posterior of the solution in a probabilistic framework. Essentially given a solution vector of coefficient weights i.e. our likelihood, we need to estimate the posterior $P(\alpha|D_{prior})$, given the prior $P(\alpha)$. We can learn this prior $P(\alpha)$ using a Kernel Density Estimation method - the Parzen window with an RBF kernel as explained below:

$$p(\alpha) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(\alpha_i - \alpha)^2}{2\sigma^2}\right) \tag{42}$$

where n is the number of prior data points, α is the estimated coefficient vector and α_i are the prior points. The bandwidth or standard deviation σ is obtained as shown in the next section.

More details about how we solve this objective function and how we learn our priors can be found in the Appendix at the end of the thesis.

4.2.8 Data Binning

One of the issues with using a density estimation technique directly is that it will give higher probability to points that occur more frequently in our prior data D_{prior}. This is not desirable as we assume that all of our prior data is accurate as we obtained it from a high accuracy 3D capture system and we want to weight them equally. So in order to overcome this, we essentially need to perform a data-binning operation where we replace multiple data points by a single vector corresponding to the mean of the cluster to which they belong. This makes sense as it allows us to give equal weight to different valid facial configurations without letting its frequency affect the probability. We use the Mean Shift algorithm [104] to cluster our prior data.

The mean shift algorithm is a non-parametric clustering algorithm which does not require the knowledge about the number of clusters. It works by updating candidates for centroids to be the mean of the points within a given region. Given a candidate centroid x_i for iteration t, the candidate is updated according to the following equation:

$$x_i^{t+1} = x_i^t + m(x_i^t) \tag{43}$$

where m is the mean shift vector that is computed for each centroid that points towards a region of the maximum increase in the density of points

We make use of an automatic method of bandwidth selection [126, 127, 128] for use with the mean shift algorithm. This gives us our bandwidth, σ in equation 42. We then use the means of the clusters thus obtained (α_i) within the Parzen window density estimation framework (Eqn. 42) in order to obtain the prior probability of a solution.

4.2.9 Objective Function with 3D Spatial Prior

Finally, we factor this probability into our objective function in order to obtain optimal coefficients α that adhere to physically plausible face configurations. Since the probability of the upper face coefficients should be independent of the lower face coefficients — e.g. to avoid the probability of the eyebrows being raised being affected by the jaw being open — we perform the density estimation of these coefficients separately. This is essential since our prior data is not guaranteed to cover all possible combinations of the upper-face and lower-face shapes in tandem. Our updated objective function is as shown below:

$$E_{Final(\alpha)} = E_{2D} + \frac{\lambda}{P(\alpha)} \tag{44}$$

where λ is the weight given to the prior term and $P(\alpha)$ is the prior on the coefficients. This ensures that the solution provided by our objective function lies within the valid space of facial expressions which is controlled by the prior data. As seen in the results in Figures 81,83,84 and 85, this yields significant improvement over the solve using just the 2D landmarks. This is especially visible in regions around the mouth as this is where majority of the variation in depth occurs.

4.2.10 Including Priors for Dynamics

So far in our markerless pipeline, we have considered solving for the Blendshape coefficient weights on a frame by frame basis. We do use a Kalman filter for online smoothing in between frames, but while this smooths out the noise in the landmark detections, it does not do so in a way that utilizes information about the dynamics of facial movements in general. Our existing data from the accurate 3D capture system not only consists information about the prior probability of coefficient weights, but also contains inherent information about how the coefficient weights change over time i.e. dynamic information. While the human face is a very complex system with many degrees of freedom, the space of human facial expressions and movements still lie within a smaller manifold within this larger space — i.e. human beings tend to perform facial expressions and speech in relatively predictable ways. It makes sense to extract this information and utilize it in order to improve the results of our solve. In order to do this, we propose a dynamic prior based on information from previous frames in the performance in order to predict the probability of the current frame, given the likelihood. We propose learning this prior using a Gaussian Process regression model.

A Gaussian Process can be thought of as a family of functions with certain properties of that family which is specified by the covariance matrix. An underlying assumption made in this case is that the family of functions is assumed to be smooth, which is a reasonable assumption to make in our case as the change in our coefficients over time is naturally smooth owing to the smooth movements of the face. This covariance matrix is a function of the existing pairs of input data points and can take the form of the family of Mercer Kernels, which are positive semidefinite. One way of thinking about this is to imagine a multivariate Gaussian distribution that is fit to our data points. Given a new input point, the value of the output can be obtained as the mean of a conditional Gaussian distribution, conditioned on the existing points and the variance is obtained similarly. As described in [129], this can be stated formally as:

The Multivariate Gaussian Distribution Theorem [130] states that – given $x = (x_1, x_2)$ is jointly Gaussian with parameters

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \Sigma_{11}\, \Sigma_{12} \\ \Sigma_{21}\, \Sigma_{22} \end{pmatrix}, \quad \Lambda = \Sigma^{-1} = \begin{pmatrix} \Lambda_{11}\, \Lambda_{12} \\ \Lambda_{21}\, \Lambda_{22} \end{pmatrix} \tag{45}$$

Then the marginals are given by

$$\begin{aligned} p(x_1) &= \mathcal{N}(x_1|\mu_1, \Sigma_{11}) \\ p(x_2) &= \mathcal{N}(x_2|\mu_2, \Sigma_{22}) \end{aligned} \tag{46}$$

and the posterior conditional is given by

$$\begin{aligned} p(x_1|x_2) &= \mathcal{N}(x_1|\mu_{1|2}, \Sigma_{1|2}) \\ \mu_{1|2} &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \\ &= \mu_1 - \Lambda_{11}^{-1}\Lambda_{12}(x_2 - \mu_2) \\ &= \Sigma_{1|2}(\Lambda_{11}\mu_1 - \Lambda_{12}((x_2 - \mu_2))) \\ \Sigma_{1|2} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} = \Lambda_{11}^{-1} \end{aligned} \tag{47}$$

Our purpose is to use Gaussian Processes for regression, so with this in mind, we'll discuss regression and show how Gaussian Processes provide a Bayesian approach to regression. Consider a model defined as a linear combination of M fixed basis functions given by the vector $\phi(x)$ so that

$$y(x) = w^T \phi(x) \tag{48}$$

where x is the input vector and w is the M-dimensional weight vector. If we consider a prior over w given by an isotropic Gaussian of the form

$$p(w) = \mathcal{N}(w|0, \alpha^{-1}\mathbf{I}) \tag{49}$$

where α represents the precision of the distribution. The probability distribution over w induces a probability distribution over $y(x)$. The joint distribution of the function values $y(x_1), ..., y(x_N)$, which we denote by $\mathbf{y}$ is given by

$$y = \Phi w \tag{50}$$

where Φ is the design matrix with elements $\Phi_{nk} = \phi_k(x_n)$. y is Gaussian distributed with mean

and covariance given by:

$$\begin{aligned} \mathbb{E}[y] &= \Phi\mathbb{E}[w] = 0 \\ cov[y] = \mathbb{E}[yy^T] &= \Phi\mathbb{E}[ww^T]\Phi^T = \frac{1}{\alpha}\Phi\Phi^T = K \end{aligned} \tag{51}$$

where K is the Gram matrix with elements

$$K_{nm} = k(x_n, x_m) \tag{52}$$

with $k(x,x')$ is the kernel function.

We also need to take into account the noise on the observed target values, given by

$$t_n = y_n + \varepsilon_n \tag{53}$$

where $y_n = y(x_n)$ and ε_n is a random noise which is Gaussian so that

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1}) \tag{54}$$

where β is a hyperparameter representing the precision of the noise. The joint distribution of the target values $t = (t_1, ..., t_N)^T$ is given by:

$$p(t|y) = \mathcal{N}(t|y, \beta^{-1}I_N) \tag{55}$$

By equation 51,

$$p(y) = \mathcal{N}(y|0, K) \tag{56}$$

The Kernel function K is typically chosen to have the property that for points that are similar (or closer), the corresponding values of $y(x_n)$ will be more strongly correlated than for dissimilar points. This ensures the condition of smoothness. The marginal distribution $p(t)$ is given by

$$p(t) = \int p(t|y)p(y)dy = \mathcal{N}(t|0, C) \tag{57}$$

where the covariance matrix C has the elements

$$C(x_n, x_m) = k(x_n, x_m) + \beta^{-1}\delta_{nm} \tag{58}$$

The Kernel function we use is the exponential of the quadratic form given by

$$k(x_n, x_m) = \theta_0 \exp\left\{ -\frac{\theta_1}{2} \|x_n - x_m\|^2 \right\} \tag{59}$$

Now given a new target variable t_{N+1} for a new input vector x_{N+1}, we evaluate the predictive distribution $p(t_{N+1}|t_N)$. From equation 57, the joint distribution over $t_1, ..., t_{N+1}$, will be given by

$$p(t_{N+1}) = \mathcal{N}(t_{N+1}|0, C_{N+1}) \tag{60}$$

where

$$C_{N+1} = \begin{pmatrix} C_N & k \\ k^T & c \end{pmatrix} \tag{61}$$

where the elements of C_N are given by equation 58, k has elements $k(x_n, x_{N+1})$ and the scalar $c = k(x_{N+1}, x_{N+1}) + \beta^{-1})$. Using the results from equation 45, the conditional distribution $p(t_{N+1}|t)$ is a Gaussian distribution with mean and covariance given by

$$m(x_{N+1}) = k^T C_N^{-1} t \tag{62}$$

$$\sigma^2(x_{N+1}) = c - k^T C_N^{-1} k \tag{63}$$

Using equations 62 and 63, we are able to predict the coefficient weights of the current frame in the animation, given the previous frames in the animation. Our input to the Gaussian Process regression model, i.e. the x_i values correspond to a window of coefficient weight vectors appended together — $\alpha_n, \alpha_{n+1}, ..., \alpha_{n+j-1}$ and the corresponding output value y_i is given by a vector of coefficient weights given by α_{n+j}. This data is obtained from our existing prior animation data (from section 4.2.7 by sliding a window of size j over the sequence of solved coefficient weights.

This gives us a prior probability for the coefficient weights α_i given the coefficient weights over the previous j frames , $\alpha_{i-j}, ..., \alpha_{i-1}$

$$\alpha_i = GP(\alpha_{(i-j):(i-1)}) \tag{64}$$

The predicted value of α_i is given by equation 62. Equation 63 gives us a measure of how confident we are about the predicted value of α_i.

Finally, we incorporate this predicted dynamic value of the coefficient weights for the current

frame into our objective function to give us

$$E_{Final(\alpha_i)} = E_{2D} + \frac{\lambda}{P(\alpha_i)} + \gamma \|\alpha_i - GP(\alpha_{(i-j):(i-1)})\|_2^2 \tag{65}$$

where γ is a weighting constant that balances the effect of the dynamic prior and is a function of the variance returned by the Gaussian Process regression — i.e. $\gamma = \frac{1}{\mathbb{V}}$, where $\mathbb{V}$ is given by equation (63).

Equation 65 has two constraints on the coefficient weights for the Blendshapes — one enforcing the prior probability given by equation (42) and the other enforcing the constraint on the dynamics which is a function of the previous j frames of the performance, given by equation (64).

4.2.11 Lighting and Texturing

In order to be able to properly relight and re-texture our 3D mesh – e.g to overlay texture on top of the video for virtual makeup – we need to optimize for the lighting parameters. This includes the intensity of the light, the position of the light and the ambient intensity of the environment. We do this using an inverse rendering operation and optimize for these parameters. The idea is to find the values for these parameters such that the squared difference in pixel intensities between the re-projected mesh vertices and the video texture is minimized. We assume a Lambertian reflection model and a single point light source.

The Lambertian rendering equation is given by:

$$I_{v_i} = I_a K_a + I_p K_d (N' \cdot L') \tag{66}$$

where:

- I_{v_i} is the resulting intensity at a vertex v_i
- I_a is the ambient light intensity,
- K_a is the ambient coefficient,
- I_p is the point-light intensity,
- K_d is the diffuse coefficient
- N' is the normalized surface normal at a vertex v_i,
- L' is the normalized light direction which depends on the light position L_{pos},

Figure 80: Results obtained after relighting, re-texturing and overlaying our 3D mesh on top of the video.

- $\cdot$ is the dot product operator

The colour C_{v_i} at a vertex v_i on the mesh is then given by

$$C_{v_i} = I_{v_i} T_{v_i}$$

where T_{v_i} is the texture colour value for the vertex v_i. Our lighting objective function then aims to to minimize the error over all the vertices on the mesh

$$E_{light(I_a,I_p,L_{pos})} = min_{I_a,I_p,L_{pos}} \sum_i ||C_{I_i} - C_{v_i}||^2$$

where C_{I_i} is the pixel colour in the original video input at which the vertex v_i is projected onto.

This gives us the optimal values for the position and intensity of the light in the scene, which allows us to reproject the 3D face back in the video using our projection matrix calculated in Section 4.2.4. The results of our relighting and re-texturing can be seen in Figure 80.

Note that we need to obtain the texture information of the actor's face beforehand in order to do this lighting optimization. In our experiments, we use the texture returned to us from the 3D scanner([10]). This has to be a diffuse texture, i.e. it cannot have shading artifacts such as shadows embedded within it. This is because the lighting in the scene can only be obtained when we have an artifact free representation of the actor's face texture, as the optimization uses the artifacts in the video as cues in order to figure out the lighting location and intensity. This can be done by having uniform illumination from all directions on the actor's face when the texture is obtained. Similarly, the video that we are solving for cannot be used as the source of the texture itself as this will mean that the optimizer will give the trivial solution where the texture values are unaltered in order to match the video, resulting in no optimization being done at all. Ghosh et al. [54] presented a method to obtain very high quality diffuse and specular albedo and normal maps of an actor's face using the Light Stage technology which, in theory, would enable one to estimate the lighting parameters with accuracy.

4.2.12 Results and Discussion

We render our 3D model within an OpenGL framework combined with OpenCV for calculation of the projection matrix and solving the perspective-n-point problem. In order to perform our optimization, we make use of the Ceres solver, an open source C++ library for modeling and solving constrained non-linear optimization problems. Our method is online as it doesn't require any operations across frames, although it isn't real-time owing to speed bottlenecks. Our system consists of a 4th generation Intel-I7 2.80GHz quadcore processor with 16GB RAM and an NVIDIA GTX765M graphics card.

While the 2D feature detector used for initializing the landmarks per frame is independent of the user (as it's trained on different faces), in our experiments it wasn't accurate enough, especially for extreme facial expressions including ones with occlusions. The Active Appearance Model is trained on 15 images of the user performing few facial expressions that elicit the range of his expressions both on the upper and lower face regions. These training images are marked semi-automatically – initialized using the previous 2D feature tracker and then corrected where needed. Although this makes our system dependent on the specific actor, it is a small price to pay for the improvement in tracking especially considering only 15 images were sufficient to track accurately through a video sequence of over a thousand frames.

The accompanying video shows the improvements obtained owing to our Kalman filtering operation performed both on the 2D features and on the camera projection parameters.

As seen in the results in Figures 81,83,84 and 85, the addition of the prior makes our results more reliable. The improvements can be seen in general but specifically around the mouth region where the changes in depth are most prominent. Areas where occlusions are common, around the eyes and mouth, stand to benefit the most as these regions are prone to loss of information in the 2D input. As our density estimator involves the Radial Basis Function as a kernel in the Parzen Window scheme, our objective function becomes non-linear, but because we initialize the parameters with the solutions from the previous frame, we do not run into popping in the animations owing to local minima. Nevertheless, the Parzen window density estimator is the prime bottleneck in taking this approach from merely online to real-time. In future work we will consider more efficient mechanisms for calculating prior probabilities such as Probabilistic PCA [85] or Gaussian Process Latent Variable models [131]. Our prior data consisted of animation sequences over 5 people and multiple facial expressions and speech movements to give us a total of 45817 frames of animation. Our automatic bandwidth estimation gives us a bandwidth of 0.3066 and 81 clusters for the lower face and a bandwidth of 0.1202 and 82 clusters for the upper face.

Finally, our lighting optimization enables us to re-texture and re-render the 3D mesh and overlay it on top of the video allowing us to augment the input video with desired textures which blend in with the user's face as can be seen in the accompanying video. In our experiments although we place no restrictions on the lighting environment, we do assume that the lighting doesn't change during the sequence, so we are able to perform the lighting optimization only on the first frame of the video and use the obtained parameters through the sequence. Of course this condition can be relaxed if we do the optimization every frame at the expense of computational efficiency.

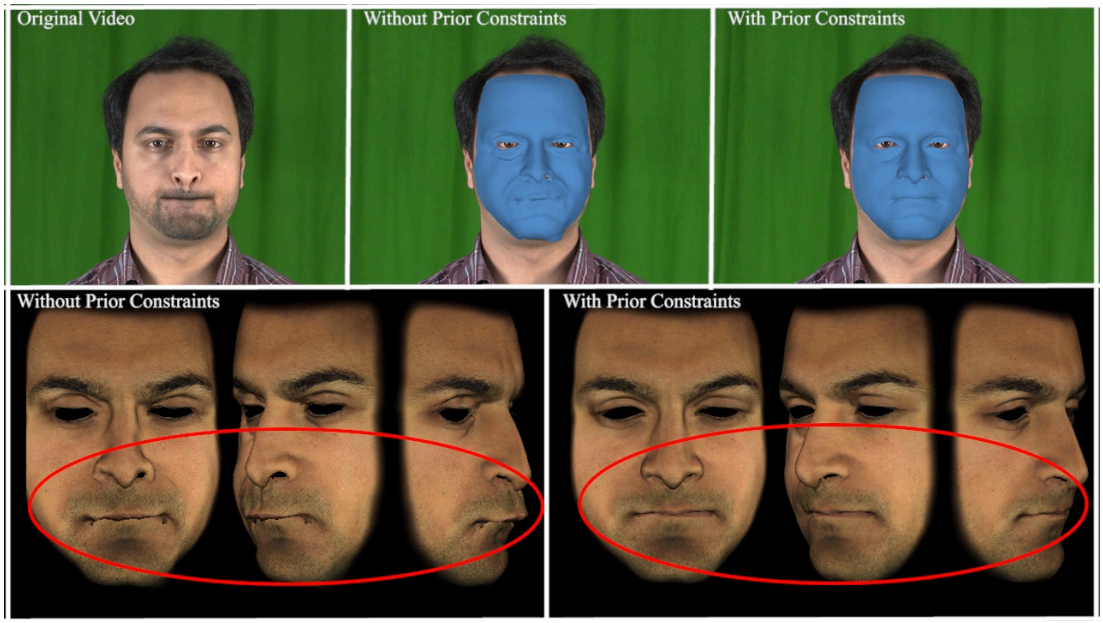


Figure 81: Comparison of a pucker expression using only 2D points vs the improvements obtained using our Prior constraint.

Figure 82: Results from our solver re-targeted onto multiple face models. As seen in the side views, our method is able to handle ambiguities in depth, inherent in monocular input, and achieve results that are physically plausible even in areas with occlusions like around the lips.

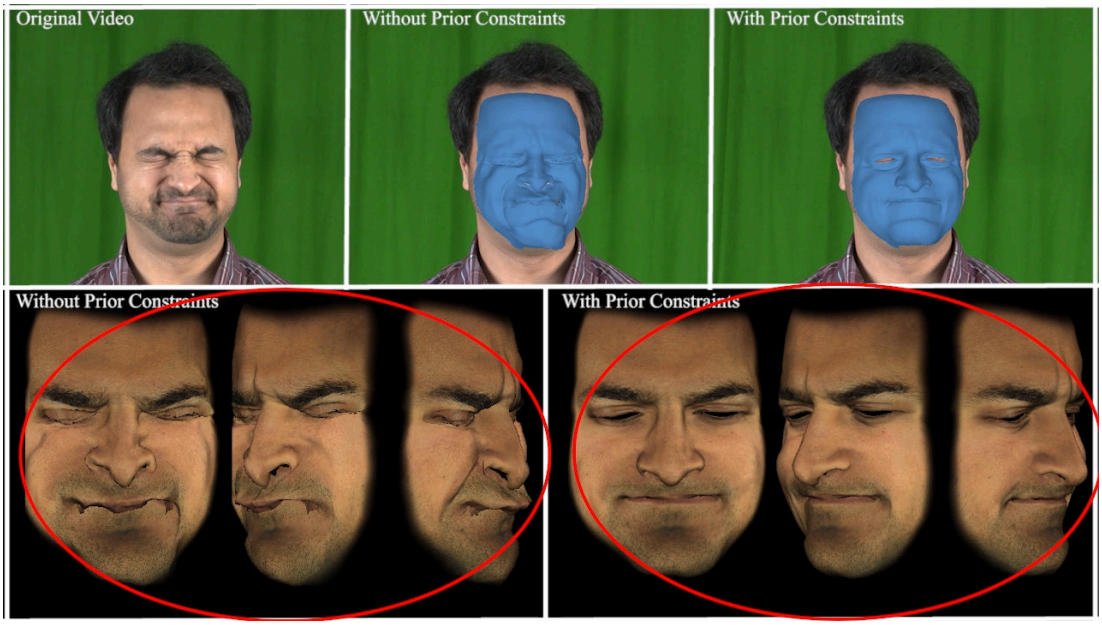


Figure 83: Comparison of a full face squeeze motion using only 2D points vs the improvements obtained using our Prior constraint.

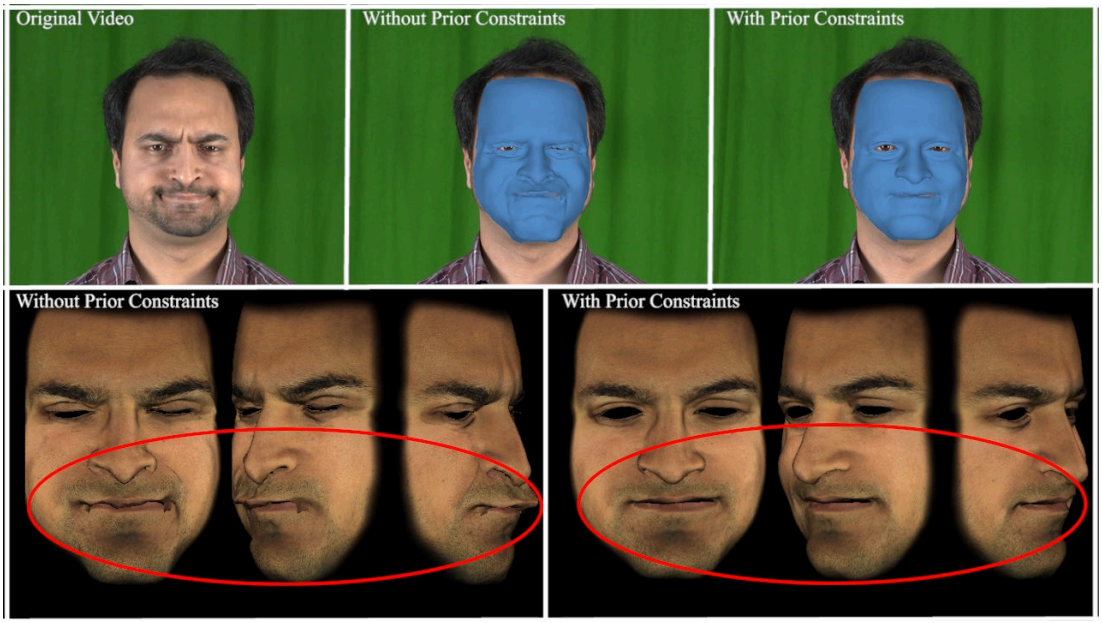


Figure 84: Comparison of angry expression using only 2D points vs the improvements obtained using our Prior constraint.

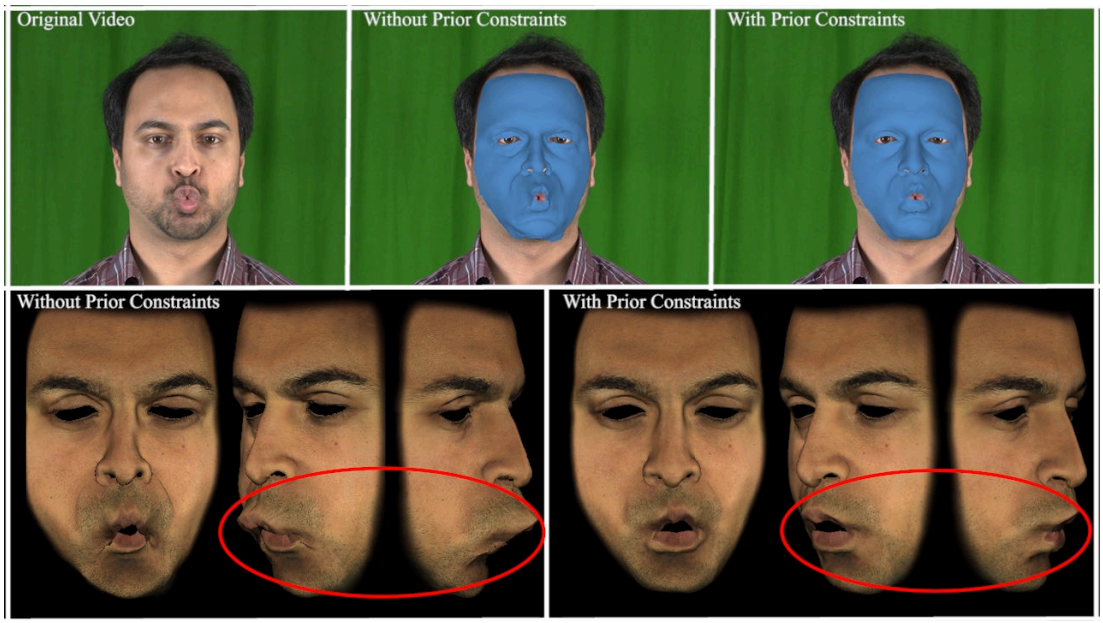


Figure 85: Comparison showing results around the mouth regions using only 2D points vs the improvements obtained using our Prior constraint.

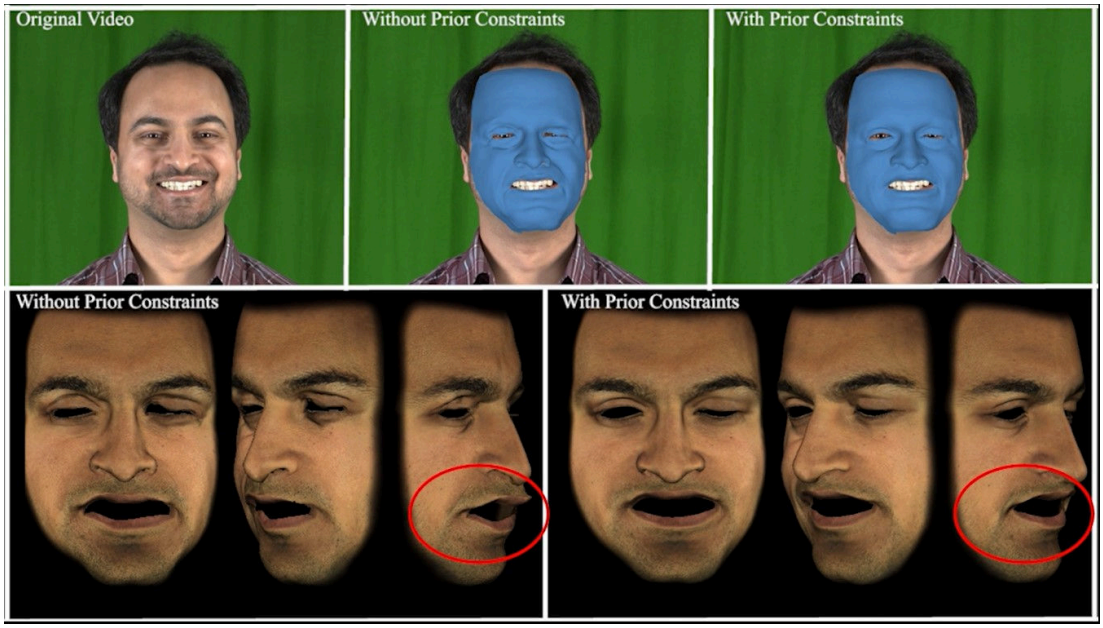


Figure 86: Comparison showing a smile expression using only 2D points vs the improvements obtained using our Prior constraint.

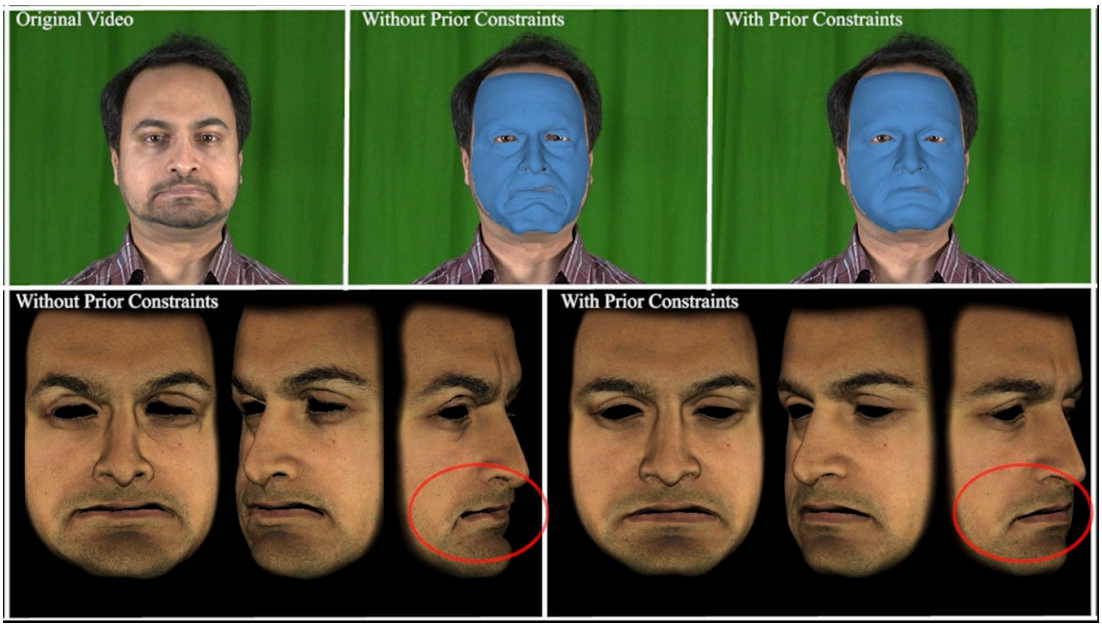


Figure 87: Comparison showing a frown expression using only 2D points vs the improvements obtained using our Prior constraint.

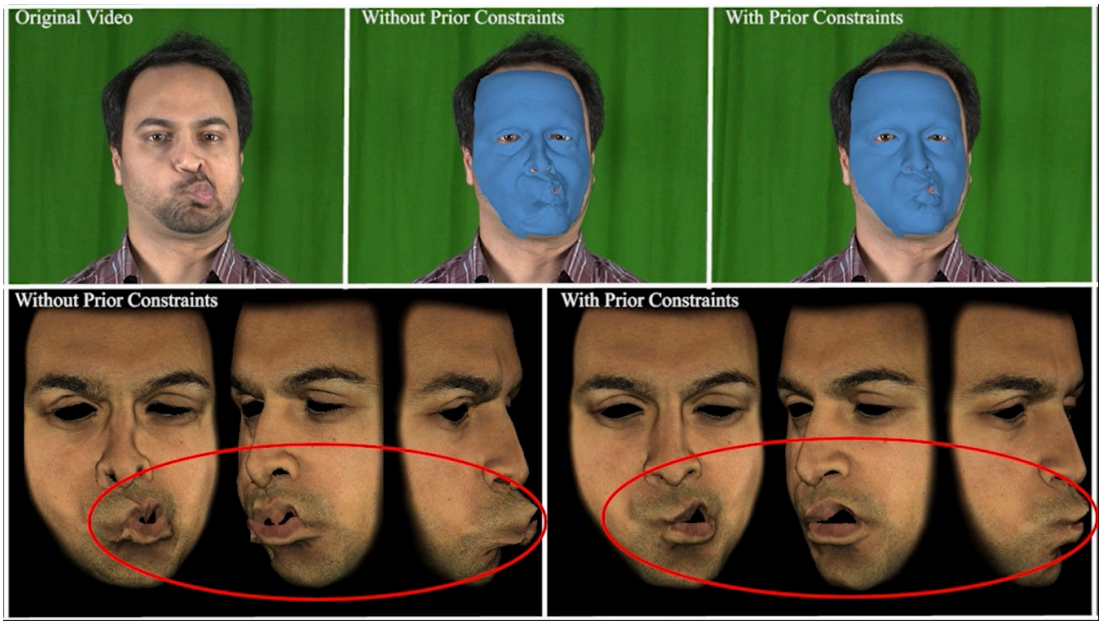


Figure 88: Comparison showing a lip-swing motion using only 2D points vs the improvements obtained using our Prior constraint.

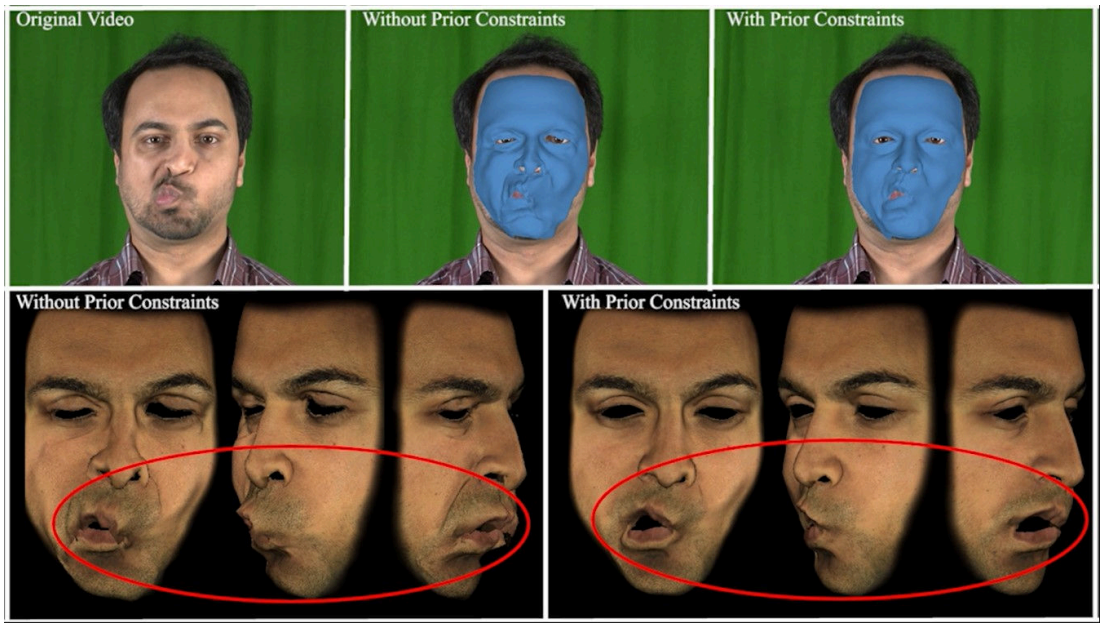


Figure 89: Comparison showing a lip-swing motion using only 2D points vs the improvements obtained using our Prior constraint.

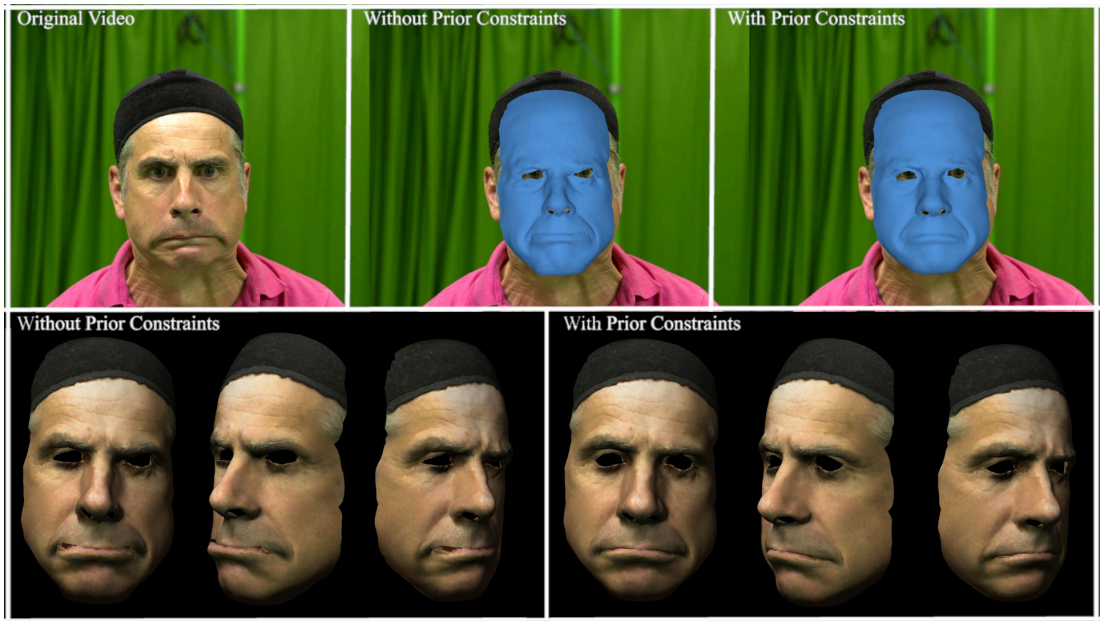


Figure 90: Comparison of the frown expression using only 2D points vs the improvements obtained using our Prior constraint.

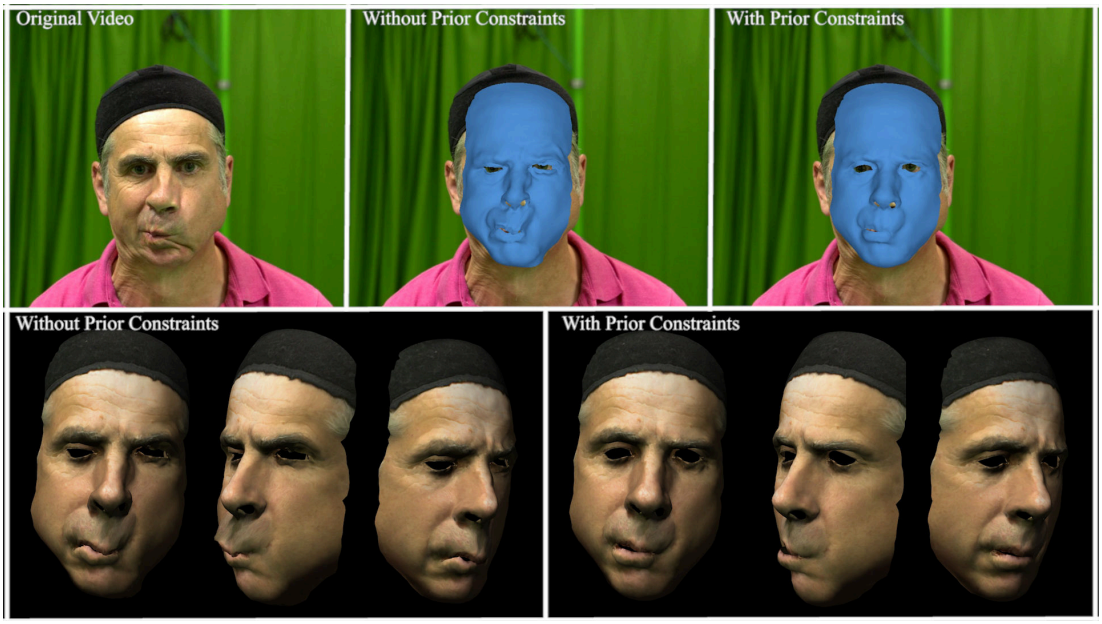


Figure 91: Comparison of the mouth region using only 2D points vs the improvements obtained using our Prior constraint.

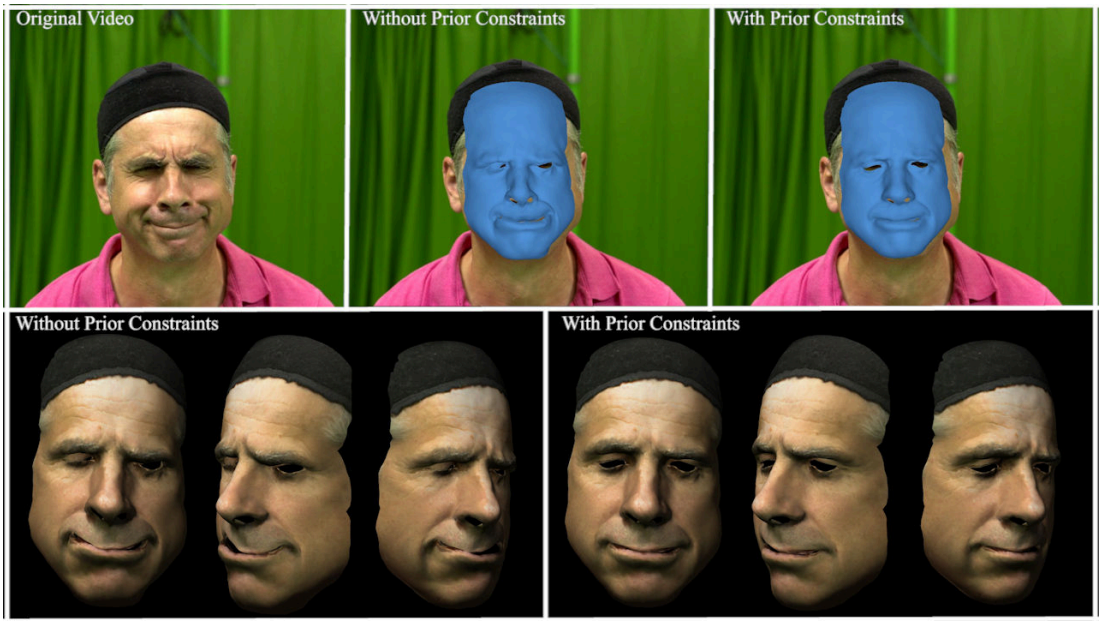


Figure 92: Comparison of results for angry expression using only 2D points vs the improvements obtained using our Prior constraint.

4.2.13 Quantitative Evaluation - Comparison with Ground Truth

In order to evaluate the results of our algorithm quantitatively, we needed to collect ground truth data for comparisons. For this purpose, we recorded one of our subjects using 6 high-definition cameras with baselines offsets that capture the actor's face from 5 different angles. These cameras were synced with each other using our gen-lock mechanism allowing us to obtain video frames from the performance sequence from different views that are temporally in correspondence with each other. We use the data from these cameras for a few expressions and reconstruct the 3D geometry of the actor's face for those frames using the Agisoft Photoscan software [132] and obtain high detail textured meshes as shown in Figure 93. The obtained meshes have a random topology and hence we use the method of Amberg et al. [11] to deform the neutral face of the actor to the reconstructed 3D meshes. This gives us ground truth geometry with the same topology as our Blendshapes which we use for quantitative evaluation. We generate a heat map based on the per-vertex error of our results compared to the ground-truth. The results of this evaluation can be seen in Figure 94. The heat map was generated by using the error value per vertex (after scaling between 0 and 1) as a UV coordinate on a color gradient image. The gradient image goes from green at 0 to red at 1. Regions that have higher error show up in red and regions with lower error show up in green. The total error from ground truth geometry for the lip-swing expression was: 4.4908e+05 units (without prior) and 3.3752e+05 (with prior). The total error from ground truth geometry for the frown expression was: 5.5391e+05 units (without prior) and 4.3288e+05 (with prior).

Figure 93: Ground-truth geometry obtained by reconstructing synchronized frames from multiple views of the actor's face.

Figure 94: Ground-truth evaluation for the lip-swing expression (top) and the frown expression (bottom). The total error from ground truth geometry for the lip-swing expression was: 4.4908e+05 units (without prior) and 3.3752e+05 (with prior). The total error from ground truth geometry for the frown expression was: 5.5391e+05 units (without prior) and 4.3288e+05 (with prior). The ground truth geometry was reconstructed from 5 different views of the actor's face. We generate a heat map based on the per-vertex error of our results compared to the ground-truth. The heat map was generated by using the error value per vertex (after scaling between 0 and 1) as a UV coordinate on a color gradient image. The gradient image goes from green at 0 to red at 1. Regions that have higher error show up in red and regions with lower error show up in green.

4.2.14 Conclusion

We have presented a lightweight markerless approach for capturing the facial movements of a user using only a 2D monocular input resulting in high quality animation parameters. We make use of prior constraints on the solve results, obtained from pre-existing accurate 3D captures and use this to improve the physical plausibility of our solve results which helps tackle the loss of information in depth that is inherent in 2D monocular inputs. We combine 2D landmark detections based on an ensemble of regression trees with an Active Appearance Model for improved accuracy. We make use of Kalman filters to handle noise across frames in an online fashion, both in 2D feature detection and in estimating the camera parameters giving us improved tracking and solves. Finally we also present our optimization for the light intensity and positions, which enables us to accurately relight and re-texture the 3D models allowing us to overlay desired textures on top of the input video.

5 Conclusions

In this thesis, we have presented a completely automated pipeline for Performance Driven Facial Capture and Animation. We have discussed the state of the art in this area and the pros and cons of various capture methods. Our work mainly makes novel contributions in both the Capture and Modeling stages of the animation pipeline. Our automated pipeline for generation of the Neutral face mesh in a desired topology from either 3D scans or from multiple images of the actor's face and the automatic generation of the Blendshape set and the actor specific nuances along with the high-frequency details such as wrinkles, significantly reduces the effort required in the Modeling stage which is significant step in the direction of consumer friendly facial animation. Our contributions to marker-based facial capture systems improves upon the state of the art by accounting for details that were previously difficult to capture using only markers, by the addition of additional features in the form of reflective patterns which are used in a FACS classification framework to improve animation results. Finally we make contributions to markerless facial capture systems using only a single monocular video input with the use of both static and dynamic priors learnt from previously captured data using more accurate capture systems, thus removing the need for costly equipment, multiple cameras and controlled studio settings.

Over the last decade, many of the pressing challenges involved in Performance Driven Facial Capture and Animation have been addressed by multiple approaches and in many cases they have been tackled effectively and efficiently. While tremendous progress has been achieved in the automation, consumerization and democratization of Performance Driven Facial Capture and Animation over the last decade, there does not exist, yet, a one-stop approach which can allow for an actor to easily be captured with high realism and fidelity, without markers or any makeup, without training phases, with free movement of the head and body at large angles to the camera, with drastically changing lighting conditions, using a single monocular camera, no user intervention with automatic acquisition and tracking of the eyes and teeth. As mentioned in the introduction section of this thesis, the ultimate goal of facial animation is a system that creates realistic animation, operates in real-time, is as automated as possible and adapts to different individual faces easily. Many of the methods over the last decade have addressed some or all of these points and have achived considerable success, even if at the cost of a trade-off between them. Most of the approaches address the overall face in general itself and do not concentrate their efforts on the eyes, teeth and hair and quite frequently, they are put in post-capture and post-retargeting by 3D artists. There have been some promising efforts at automatically generating meshes and texture along with reflectance data for the eyes as seen in Beeler et al. 2014 [133], but the work addresses static reconstruction and deformation of the

iris while the dynamic aspects of the eyeball and its complex motion patterns are not addressed. Similarly, the teeth are usually added in with the help of a 3D artist after the capture session and not obtained as a part of the capture pipeline. Thies et al. [40] re-render the teeth in the interior region of the face in their re-targeted and re-rendered image but it is only a 2D approximation as opposed to a realistic 3D acquisition which can be animated along with the mouth. Luo et al. [134] presented an approach for synthesizing realistic hair in 2013, but it still hasn't been organically incorporated into the facial capture and animation pipeline and stands as a separate system.

Most of the work in facial animation has either made an assumption of little or no motion of the face with respect to the cameras. To our knowledge, none of the approaches allow for fully free motion of the actor and the actor's face spanning multiple large angles, drastically changing lighting and large motions of the actor with respect to the camera. Head Mounted Systems allow the actor to move but the location and angle with respect to the camera is still fixed. An interesting and useful direction to pursue in the future will be to allow for fully free motion of the actor without a Head Mounted Device, without any markers or special makeup and using multiple cameras. Using multiple cameras and automatic feature detection it would be possible to track the actor's face and combine information from all the cameras in order to track the actor with completely free movement of the head and body. There is also lot of information about identity of an individual encoded in the dynamics of the facial expression as recently shown by Girges et al. [135]. Another promising future direction would be in learning the actor specific dynamics of facial motion and using this to modify a performance by another actor in order to simulate identity.

References

[1] D. Bennett, "The faces of the polar express," in *ACM Siggraph 2005 Courses*, p. 6, ACM, 2005.

[2] J. L. F Pighin, "Performance-driven facial animation," 2006.

[3] O. Alexander, M. Rogers, W. Lambeth, J.-Y. Chiang, W.-C. Ma, C.-C. Wang, and P. Debevec, "The digital emily project: Achieving a photorealistic digital actor," *IEEE Computer Graphics and Applications*, vol. 30, no. 4, pp. 20–31, 2010.

[4] J. Von der Pahlen, J. Jimenez, E. Danvoye, P. Debevec, G. Fyffe, and O. Alexander, "Digital ira and beyond: creating real-time photoreal digital actors," in *ACM SIGGRAPH 2014 Courses*, p. 1, ACM, 2014.

[5] T. Beeler, F. Hahn, D. Bradley, B. Bickel, P. Beardsley, C. Gotsman, R. W. Sumner, and M. Gross, "High-quality passive facial performance capture using anchor frames," *ACM Trans. Graph*, vol. 30, no. 4, 2011.

[6] E. Sifakis, I. Neverov, and R. Fedkiw, "Automatic determination of facial muscle activations from sparse motion capture marker data," in *Acm transactions on graphics (tog)*, vol. 24, pp. 417–425, ACM, 2005.

[7] Vicon, "Vicon," *http://www.vicon.com*.

[8] D. Bradley, W. Heidrich, T. Popa, and A. Sheffer, "High resolution passive facial performance capture," *ACM Trans. Graph.*, vol. 29, no. 4, 2010.

[9] N. Kholgade, I. Matthews, and Y. Sheikh, "Content retargeting using parameter-parallel facial layers," *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, p. 195 204, 2011.

[10] Artec, "Artec 3d scanners," *http://www.artec3d.com*.

[11] B. Amberg, S. Romdhani, and T. Vetter, "Optimal step nonrigid icp algorithms for surface registration," *In Proc. of IEEE Computer Vision and Pattern Recognition (CVPR)*, 2007.

[12] H. Li *et al.*, "Reconstruction using structured light," *Universitat Karlsruhe (TH) Fakultat fur Informatik, Institut fur Betriebs-und Dialogsysteme, Prof. Dr. H. Prautzsch, Undergraduate Research Project,(Studienarbeit) Feb*, vol. 23, p. 64, 2004.

[13] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[14] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic, "Incremental face alignment in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1859–1866, 2014.

[15] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 25, no. 8, pp. 930–943, 2003.

[16] R. W. Sumner and J. Popovic, "Deformation transfer for triangle meshes," *ACM Trans. Graph*, vol. 23, no. 3, 2004.

[17] H. Li, T. Weise, and M. Pauly, "Example-based facial rigging," *ACM Trans. Graph*, vol. 29, no. 4, 2010.

[18] F. I. Parke, "Computer generated animation of faces," in *Proceedings of the ACM annual conference-Volume 1*, pp. 451–457, ACM, 1972.

[19] Y. Lanir, "Skin mechanics," *Handbook of Bioengineering*, pp. 11–1, 1987.

[20] M. Mori, K. F. MacDorman, and N. Kageki, "The uncanny valley [from the field]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 98–100, 2012.

[21] P. Havaldar, "Sony pictures imageworks," in *ACM SIGGRAPH 2006 Courses*, p. 5, ACM, 2006.

[22] J.-y. Noh and U. Neumann, "A survey of facial modeling and animation techniques," tech. rep., USC Technical Report, 99–705, 1998.

[23] C. Bregler, "Motion capture technology for entertainment [in the spotlight]," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 160–158, 2007.

[24] L. Williams, "Performance-driven facial animation," *ACM SIGGRAPH*, vol. 24, no. 4, pp. 235–242, 1990.

[25] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin, "Making faces," *Proc. 25th Annu. Conf. Comput. Graph. Interact. Tech*, 1998.

[26] G. Borshukov, D. Piponi, O. Larsen, J. P. Lewis, and C. Tempelaar-lietz, "Universal capture - image-based facial animation for the matrix reloaded," *ACM SIGGRAPH Courses*, 2005.

[27] G. Fyffe, T. Hawkins, C. Watts, W.-C. Ma, and P. Debevec, "Comprehensive facial performance capture," *Computer Graphics Forum*, vol. 30, no. 2, p. 425434, 2011.

[28] G. Fyffe, A. Jones, O. Alexander, R. Ichikari, and P. Debevec, "Driving high-resolution facial scans with video performance capture," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 1, p. 8, 2014.

[29] T. F. Cootes, G. J. Edwards, , and C. J. Taylor, "Active appearance models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, p. 681685, 2001.

[30] H. Li, J. Yu, Y. Ye, and C. Bregler, "Realtime facial animation with on-the-fly correctives," *ACM Trans. Graph.*, vol. 32, no. 4, 2013.

[31] J. R. Tena, F. D. la Torre, and I. Matthews, "Interactive region-based linear 3d face models," *ACM Trans. Graph.*, vol. 30, no. 4, 2011.

[32] J. Booth, A. Roussos, S. Zafeiriou, A. Ponniah, and D. Dunaway, "A 3d morphable model learnt from 10,000 faces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5543–5552, 2016.

[33] C. Cao, Y. Weng, S. Lin, and K. Zhou, "3d shape regression for real-time facial animation," *ACM Tran. Grap.*, vol. 32, no. 4, 2013.

[34] C. Cao, Q. Hou, and K. Zhou, "Displaced dynamic expression regression for real-time facial tracking and animation," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 43, 2014.

[35] C. Cao, D. Bradley, K. Zhou, and T. Beeler, "Real-time high-fidelity facial performance capture," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 46, 2015.

[36] P. Garrido, L. Valgaerts, C. Wu, and C. Theobalt, "Reconstructing detailed dynamic face geometry from monocular video," *ACM Trans. Graph*, vol. 32, no. 6, 2013.

[37] S. Bouaziz, Y. Wang, and M. Pauly, "Online modeling for realtime facial animation," *ACM Trans. Graph*, vol. 32, no. 4, 2013.

[38] Z. Deng, P.-Y. Chiang, P. Fox, and U. Neumann, "Animating blendshape faces by cross-mapping motion capture data," *Proc. of symposium on Interactive 3D graphics and games (SI3D)*, 2006.

[39] K. S. Bhat, R. Goldenthal, Y. Ye, R. Mallet, and M. Koperwas, "High fidelity facial animation capture and retargeting with contours," *Proc. 12th ACM SIGGRAPH/Eurographics Symp. Comput. Animat. (SCA)*, 2013.

[40] J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt, "Real-time expression transfer for facial reenactment," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 183, 2015.

[41] K. Venkataraman, S. Lodha, and R. Raghavan, "A kinematic-variational model for animating skin with wrinkles," *Computers & Graphics*, vol. 29, no. 5, pp. 756–770, 2005.

[42] Y. Wu, P. Kalra, and N. M. Thalmann, "Simulation of static and dynamic wrinkles of skin," in *Computer Animation'96. Proceedings*, pp. 90–97, IEEE, 1996.

[43] E. Sifakis, A. Selle, A. Robinson-Mosher, and R. Fedkiw, "Simulating speech with a physics-based facial muscle model," in *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 261–270, Eurographics Association, 2006.

[44] D. Terzopoulos and K. Waters, "Analysis and synthesis of facial image sequences using physical and anatomical models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 569–579, 1993.

[45] S. Morishima, T. Ishikawa, and D. Terzopoulos, "Facial muscle parameter decision from 2d frontal image," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, vol. 1, pp. 160–162, IEEE, 1998.

[46] B. Choe, H. Lee, and H.-S. Ko, "Performance-driven muscle-based facial animation," *Computer Animation and Virtual Worlds*, vol. 12, no. 2, pp. 67–79, 2001.

[47] K. Kähler, J. Haber, and H.-P. Seidel, "Geometry-based muscle modeling for facial animation," in *Graphics interface*, vol. 2001, pp. 37–46, 2001.

[48] G. Borshukov, J. Montgomery, and W. Werner, "Playable universal capture: compression and real-time sequencing of image-based facial animation," in *ACM SIGGRAPH 2006 Courses*, p. 8, ACM, 2006.

[49] Y. Seol, J. Lewis, J. Seo, B. Choi, K. Anjyo, and J. Noh, "Spacetime expression cloning for blendshapes," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 2, p. 14, 2012.

[50] I.-C. Lin and M. Ouhyoung, "Mirror mocap: Automatic and efficient capture of dense 3d facial motion parameters from video," *The Visual Computer*, vol. 21, no. 6, pp. 355–372, 2005.

[51] B. Bickel, M. Botsch, R. Angst, W. Matusik, M. Otaduy, H. Pfister, and M. Gross, "Multi-scale capture of facial geometry and motion," *ACM Trans. Graph.*, vol. 26, no. 3, 2007.

[52] L. Zhang, N. Snavely, B. Curless, and S. M. Seitz, "Spacetime faces: High-resolution capture for modeling and animation," in *Data-Driven 3D Facial Animation*, pp. 248–276, Springer, 2008.

[53] T. Weise, H. Li, L. V. Gool, and M. Pauly, "Face-off: Live facial puppetry," *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer animation*, 2009.

[54] A. Ghosh, G. Fyffe, B. Tunwattanapong, J. Busch, X. Yu, and P. Debevec, "Multi-view face capture using polarized spherical gradient illumination," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 6, p. 129, 2011.

[55] Y. Wang, X. Huang, C.-S. Lee, S. Zhang, Z. Li, D. Samaras, D. Metaxas, A. Elgammal, and P. Huang, "High resolution acquisition, learning and transfer of dynamic 3-d facial expressions," in *Computer Graphics Forum*, vol. 23, pp. 677–686, Wiley Online Library, 2004.

[56] L. Valgaerts, C. Wu, A. Bruhn, H.-P. Seidel, and C. Theobalt, "Lightweight binocular facial performance capture under uncontrolled lighting.," *ACM Trans. Graph.*, vol. 31, no. 6, p. 187, 2012.

[57] H. Pyun, Y. Kim, W. Chae, H. W. Kang, and S. Y. Shin, "An example-based approach for facial expression cloning," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 167–176, Eurographics Association, 2003.

[58] J.-y. Noh and U. Neumann, "Expression cloning," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 277–288, ACM, 2001.

[59] C. Curio, M. Breidt, M. Kleiner, Q. C. Vuong, M. A. Giese, and H. H. Bülthoff, "Semantic 3d motion retargeting for facial animation," in *Proceedings of the 3rd symposium on Applied perception in graphics and visualization*, pp. 77–84, ACM, 2006.

[60] T. Costigan, M. Prasad, and R. McDonnell, "Facial retargeting using neural networks," in *Proceedings of the Seventh International Conference on Motion in Games*, pp. 31–38, ACM, 2014.

[61] **S. Ravikumar**, C. Davidson, D. Kit, N. Campbell, L. Benedetti, and D. Cosker, "Reading between the dots: Combining 3d markers and facs classification for high-quality blendshape facial animation.," in *Graphics Interface*, pp. 143–151, 2016.

[62] J. Serra, O. Cetinaslan, **S. Ravikumar**., V. Orvalho, and D. Cosker, "Easy generation of facial animation using motion graphs," *Computer Graphics Forum*, pp. 1467–8659, 2017.

[63] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[64] P. Ekman, W. Friesen, and C. Hager, "The facial action coding system," 2002.

[65] F. I. Parke, "A parametric model for human faces.," tech. rep., UTAH UNIV SALT LAKE CITY DEPT OF COMPUTER SCIENCE, 1974.

[66] S. M. Platt and N. I. Badler, "Animating facial expressions," in *ACM SIGGRAPH computer graphics*, vol. 15, pp. 245–252, ACM, 1981.

[67] P. Ekman and W. V. Friesen, "Measuring facial movement," *Journal of Nonverbal Behavior*, vol. 1, no. 1, pp. 56–75, 1976.

[68] K. Waters, "A muscle model for animation three-dimensional facial expression," *Acm siggraph computer graphics*, vol. 21, no. 4, pp. 17–24, 1987.

[69] J. P. Lewis and F. I. Parke, "Automated lip-synch and speech synthesis for character animation," in *ACM SIGCHI Bulletin*, vol. 17, pp. 143–147, ACM, 1987.

[70] M. M. Cohen, D. W. Massaro, *et al.*, "Modeling coarticulation in synthetic visual speech," *Models and techniques in computer animation*, vol. 92, pp. 139–156, 1993.

[71] O. Alexander, J. Busch, P. Graham, B. Tunwattanapong, A. Jones, K. Nagano, R. Ichikari, P. Debevec, and G. Fyffe, "Digital ira: High-resolution facial performance playback," 2013.

[72] B. DIGITIZER, "4012/pd (1989) from cyberware laboratory inc," *Monterey, CA*.

[73] B. Bickel, M. Lang, M. Botsch, M. A. Otaduy, and M. Gross, "Pose-space animation and transfer of facial details," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 57–66, Eurographics Association, 2008.

[74] W.-C. Ma, A. Jones, J.-Y. Chiang, T. Hawkins, S. Frederiksen, P. Peers, M. Vukovic, M. Ouhyoung, and P. Debevec, "Facial performance synthesis using deformation-driven polynomial displacement maps," in *ACM Transactions on Graphics (TOG)*, vol. 27, p. 121, ACM, 2008.

[75] W.-C. Ma, T. Hawkins, P. Peers, C.-F. Chabert, M. Weiss, and P. Debevec, "Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination," in *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pp. 183–194, Eurographics Association, 2007.

[76] H. Huang, J. Chai, X. Tong, and H.-T. Wu, "Leveraging motion capture and 3d scanning for high-fidelity facial performance acquisition," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, p. 74, 2011.

[77] M. J. Black and Y. Yacoob, "Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion," in *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pp. 374–381, IEEE, 1995.

[78] I. Essa, S. Basu, T. Darrell, and A. Pentland, "Modeling, tracking and interactive animation of faces and heads//using input from video," in *Computer Animation'96. Proceedings*, pp. 68–79, IEEE, 1996.

[79] D. Terzopoulos, A. Witkin, and M. Kass, "Constraints on deformable models: Recovering 3d shape and nonrigid motion," *Artificial intelligence*, vol. 36, no. 1, pp. 91–123, 1988.

[80] D. DeCarlo and D. Metaxas, "The integration of optical flow and deformable models with applications to human face shape and motion estimation," in *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pp. 231–238, IEEE, 1996.

[81] F. Pighin, R. Szeliski, and D. Salesin, "Resynthesizing facial animation through 3d model-based tracking," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 1, p. 143150, 1999.

[82] E. Chuang and C. Bregler, "Performance driven facial animation using blendshape interpolation," *Computer Science Technical Report, Stanford University*, vol. 2, no. 2, p. 3, 2002.

[83] J.-x. Chai, J. Xiao, and J. Hodgins, "Vision-based control of 3d facial animation," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 193–206, Eurographics Association, 2003.

[84] D. Vlasic, M. Brand, H. Pfister, and J. Popovic, "Face transfer with multilinear models," *ACM Trans. Grap.*, vol. 24, no. 3, 2005.

[85] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[86] D. Cristinacce and T. F. Cootes, "Feature detection and tracking with constrained local models.," in *BMVC*, vol. 1, p. 3, 2006.

[87] M. Lau, J. Chai, Y.-Q. Xu, and H.-Y. Shum, "Face poser: interactive modeling of 3d facial expressions using model priors," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 161–170, Eurographics Association, 2007.

[88] Y. Furukawa and J. Ponce, "Dense 3d motion capture from synchronized video streams," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, IEEE, 2008.

[89] Y. Furukawa and J. Ponce, "Dense 3d motion capture for human faces," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1674–1681, IEEE, 2009.

[90] J. M. Saragih, S. Lucey, and J. F. Cohn, "Real-time avatar animation from a single image," in *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pp. 117–124, IEEE, 2011.

[91] A. H. Bermano, D. Bradley, T. Beeler, F. Zund, D. Nowrouzezahrai, I. Baran, O. Sorkine-Hornung, H. Pfister, R. W. Sumner, B. Bickel, *et al.*, "Facial performance enhancement using dynamic shape space analysis," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 2, p. 13, 2014.

[92] F. Shi, H.-T. Wu, X. Tong, and J. Chai, "Automatic acquisition of high-fidelity facial performances using monocular videos," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 6, p. 222, 2014.

[93] S. Suwajanakorn, I. Kemelmacher-Shlizerman, and S. Seitz, "Total moving face reconstruction," *ECCV*, vol. 8692, pp. 796–812, 2014.

[94] I. Kemelmacher-Shlizerman and S. M. Seitz, "Face reconstruction in the wild," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 1746–1753, IEEE, 2011.

[95] H. Li, B. Adams, L. J. Guibas, and M. Pauly, "Robust single-view geometry and motion reconstruction," in *ACM Transactions on Graphics (TOG)*, vol. 28, p. 175, ACM, 2009.

[96] T. Weise, S. Bouaziz, H. Li, and M. Pauly, "Realtime performance-based facial animation," in *ACM Transactions on Graphics (TOG)*, vol. 30, p. 77, ACM, 2011.

[97] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," *ACM SIGGRAPH*, p. 187194, 1999.

[98] T. Baltrušaitis, P. Robinson, and L.-P. Morency, "3d constrained local model for rigid and non-rigid facial tracking," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2610–2617, IEEE, 2012.

[99] I. Metrics, "Image metrics live driver," *http://image-metrics.com*.

[100] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. Pighin, and Z. Deng, "Practice and Theory of Blendshape Facial Models," in *Eurographics 2014 - State of the Art Reports* (S. Lefebvre and M. Spagnuolo, eds.), The Eurographics Association, 2014.

[101] V. Orvalho, F. Parke, and X. Alvarez, "A facial rigging survey," *Proc. Eurographics*, vol. 32, pp. 10–32, 2012.

[102] J. M. Saragih, S. Lucey, and J. F. Cohn, "Face alignment through subspace constrained mean-shifts," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1034–1041, IEEE, 2009.

[103] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1867–1874, 2014.

[104] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on information theory*, vol. 21, no. 1, pp. 32–40, 1975.

[105] W. Wenchao, E. E. Kuruoglu, W. Shilin, L. Shenghong, and L. Jianhua, "Automatic lip contour extraction using both pixel-based and parametric models," *Ł*, vol. 22, no. CJE-1, pp. 76–82, 2013.

[106] T. Beeler, B. Bickel, P. Beardsley, B. Sumner, and M. Gross, "High-quality single-shot capture of facial geometry," *ACM Trans. Graph.*, vol. 29, no. 4, 2010.

[107] H. Li, R. W. Sumner, and M. Pauly, "Global correspondence optimization for non-rigid registration of depth scans," in *Proc. of the Symposium on Geometry Processing*, pp. 1421–1430, 2008.

[108] J. Radford, "Virtual history: the secret plot to kill hitler," in *ACM SIGGRAPH 2006 Courses*, p. 6, ACM, 2006.

[109] M. Sagar, "Facial performance capture and expressive translation for king kong," in *ACM SIGGRAPH 2006 Sketches*, p. 26, ACM, 2006.

[110] R. W. T. H. M. K. Kiran Bhat, Pablo Helman, "Industrial light and magic presents - capturing the teenage mutant ninja turtles," 2014.

[111] B. Cantwell, P. Warner, M. Koperwas, and K. Bhat, "Ilm facial performance capture," in *ACM SIGGRAPH 2016 Talks*, p. 26, ACM, 2016.

[112] S. Bouaziz, *Realtime Face Tracking and Animation*. PhD thesis, 2015.

[113] "Dimensional-imaging," *http://www.di4d.com*.

[114] B. Jiang, M. F. Valstar, and M. Pantic, "Facial action detection using block-based pyramid appearance descriptors," in *Proceedings of the ASE/IEEE International Conference on Social Computing (SocialCom 2012)*, (Amsterdam, The Netherlands), September 2012.

[115] M. F. Valstar and M. Pantic, "Combined support vector machines and hidden markov models for modeling facial action temporal dynamics," in *IEEE Int'l Conf. Computer Vision (ICCV'07), Workshop on Human Computer Interaction (HCI'07)*, pp. 118–127, 2007.

[116] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 679–698, 1986.

[117] C. Tomasi and T. Kanade, *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991.

[118] W. Li, D. Cosker, M. Brown, and R. Tang, "Optical flow estimation using laplacian mesh energy," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2435–2442, 2013.

[119] Raitt, "Presentation at u. southern california institute for creative technologiess frontiers of facial animation workshop," 2004.

[120] M. Contour, "Mova contour," *http://www.mova.com*.

[121] G. Tzimiropoulos and M. Pantic, "Optimization problems for fast aam fitting in-the-wild," in *Proceedings of the IEEE international conference on computer vision*, pp. 593–600, 2013.

[122] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[123] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: Database and results," *Image and Vision Computing*, vol. 47, pp. 3–18, 2016.

[124] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "A semi-automatic methodology for facial landmark annotation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 896–903, 2013.

[125] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: The first facial landmark localization challenge," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 397–403, 2013.

[126] J. E. Chacón and T. Duong, "Bandwidth selection for multivariate density derivative estimation, with applications to clustering and bump hunting," tech. rep., 2012.

[127] I. Horová, J. Koláček, and K. Vopatová, "Full bandwidth matrix selectors for gradient kernel density estimate," *Computational Statistics & Data Analysis*, vol. 57, no. 1, pp. 364–376, 2013.

[128] J. E. Chacón and P. Monfort, "A comparison of bandwidth selectors for mean shift clustering," *arXiv preprint arXiv:1310.7855*, 2013.

[129] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[130] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[131] N. D. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data," *Advances in neural information processing systems*, vol. 16, no. 3, pp. 329–336, 2004.

[132] L. AgiSoft and R. St Petersburg, "Agisoft photoscan," *Professional Edition*, 2014.

[133] P. Bérard, D. Bradley, M. Nitti, T. Beeler, and M. H. Gross, "High-quality capture of eyes.," *ACM Trans. Graph.*, vol. 33, no. 6, pp. 223–1, 2014.

[134] L. Luo, H. Li, and S. Rusinkiewicz, "Structure-aware hair capture," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 76, 2013.

[135] C. Girges, J. Spencer, and J. O'Brien, "Categorizing identity from facial motion," *The Quarterly Journal of Experimental Psychology*, vol. 68, no. 9, pp. 1832–1843, 2015.

Appendices

Learning The Prior

Given a set of Blendshapes and the 3D marker locations every frame obtained from the Cara system, we learn our prior by first solving for coeffecient weights x. The basic equation we are trying to optimize for is a quadratic of the following form.

$$\begin{aligned} &(Basis * x - target)^2 \\ &= (b * x - t)^T * (b * x - t) \\ &= x^T * (b^T * b) * x - 2 * (t^T * b) * x + t^T * t \end{aligned} \tag{67}$$

Where:

- b (Basis) is a matrix of size $3n \times N$, that contains the deltas for each of the Blendshapes $B_{i...N}$, where N is the number of Blendshapes and n is the number of markers.
- x is a $N \times 1$ vector of weights with the constraint $0 \leq x_i \leq 1$.
- t (target) is a $3n \times 1$ vector representing the target marker locations.

Now Matlab's quadratic programming method solves for x in the following equation

$$min_x \ \frac{1}{2} x^T H x + f^T x \quad such\ that \quad \begin{cases} A \cdot x \leq B, \\ Aeq \cdot x = beq, \\ lb \leq x \leq ub \end{cases} \tag{68}$$

H, A, and Aeq are matrices, and f, b, beq, lb, ub, and x are vectors.

A and B allow us to specify the inequality on x i.e. $0 \leq x \leq 1$

Aeq and beq are not needed as we don't have any equality constraints.

So in our case, if we compare equation 68 and equation 67, we find that $H = b^T * b + \varepsilon * I$, where I is an $N \times N$ identity matrix and ε is a very small value used for regularization.
Also according to our equation, $f = b^T * t$

Now the way we add in the L1 norm on x, i.e. $\beta\|x\|_1$, is as follows:
We add the term $\beta * 1$ to f i.e. $f = f + \beta * 1$ where 1 is a $N \times 1$ vector.

This works because in equation 68, f^T gets multiplied by x. So the value β which we add into f is multiplied by the Blendshape weights x and added to the error. Because our values of x are strictly positive, this acts as an L1-norm, effectively penalizing by a value of $\beta * x_i$ for each Blendshape weight x_i that is added. This penalizes the use of excessive weights.

The above equations are used to calculate the coeffiecients in the 3D case, which we use to learn the prior on the coefficients.

Equation (33) essentially does exactly what is described above. α in equation (33) corresponds to x in the discussion above.

We solve using the above method for every frame in our existing data (from the Cara system) and obtain multiple x values for each frame. We then perform the mean shift operation on all the obtained x vectors in order to cluster the data points and keep only the means, which we use in the next step.